ISSN 0265-2919

80p

17

THE HOME COMPUTER ADVANCED COURSE

MAKING THE MOST OF YOUR MICRO

An ORBIS Publication

IR £1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

CONTENTS

Next Week

- **We review a selection of lap held micros. These are lighter and therefore more versatile than the portable micros we have previously looked at.**
- **The Advance 86 is one of the new generation of 16-bit computers. Although it is sold as a home computer, it can be upgraded to a sophisticated business computer with software compatibility with the IBM PC.**
- **Continuing our course on improving your BASIC programming, we look at the importance of documenting your programs and suggest some rules to follow.**

QUIZ

1) What is the function of a print server?

2) What function does the command CHR$(13) accomplish?

3) What are the two standard types of video output for home computers?

4) Why is it necessary to store the contents of the registers when entering a machine code subroutine? Where are they stored?

Answers To Last Week's Quiz

A1) Manufacturers' speeds do not take account of character changes or carriage returns.

A2) Brute force in computers means comparing the advantages of as many moves as possible within the time allowed.

A3) The call addresses are: for the Spectrum hex10, for the Commodore 64 hexFFEE and for the BBC hexFFD2.

A4) In early 1984, Dragon Data announced that it was developing an MSX standard machine, in common with Japanese manufacturers.

QUIZ

COVER PHOTOGRAPHY BY PAUL CHAVE

Editor Jim Lennox; **Art Director** David Whelan; **Technical Editor** Brian Morris; **Production Editor** Catherine Cardwell; **Picture Editor** Claudia Zeff; **Sub Editor** Robert Pickering; **Designer** Julian Dorr; **Art Assistant** Liz Dixon; **Editorial Assistant** Stephen Malone; **Assistant Sub Editor** Steve Mann; **Contributors** Steven Colwill, Matt Nicholson, Mike Wesley, Geoff Nairn, Tony Harrington, Geoff Bains, Graham Storrs, Richard Pawson; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Chris Cooper; **Production Co-ordinator** Ian Paton; **Circulation Director** David Breed; **Marketing Director** Michael Joyce; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 85 Charlotte Street, London W1P 1LB; **© APSIF Copenhagen 1984; © Orbis Publishing Ltd 1984: Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Artisan Press Ltd, Leicester**

HOME COMPUTER ADVANCED COURSE – Price UK 80p IR £1.00 AUS $1.95 NZ $2.25 SA R1.95 SINGAPORE $4.50 USA and CANADA $1.95

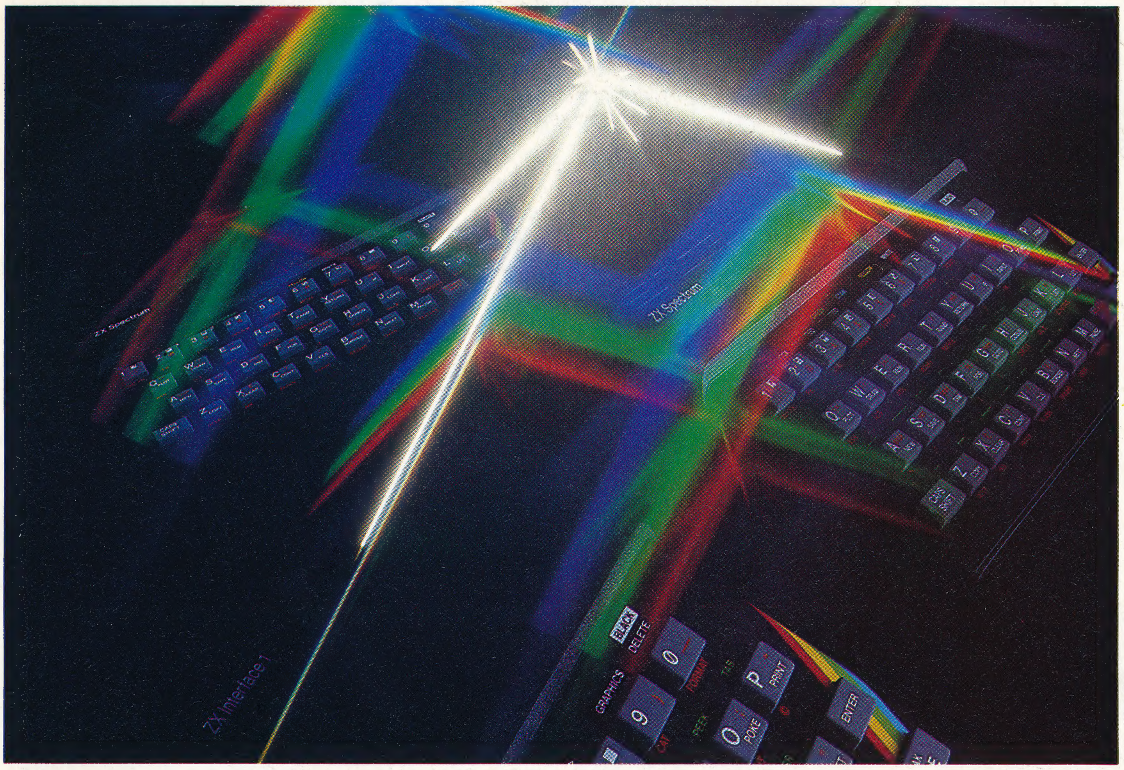
How to obtain your copies of HOME COMPUTER ADVANCED COURSE – Copies are obtainable by placing a regular order at your newsagent, or by taking out a subscription. Subscription rates: for six months (26 issues) £23.80; for one year (52 issues) £47.60. Send your order and remittance to Punch Subscription Services, Watling Street, Bletchley, Milton Keynes, Bucks MK2 2BW, being sure to state the number of the first issue required.

Back Numbers UK and Eire – Back numbers are obtainable from your newsagent or from HOME COMPUTER ADVANCED COURSE. Back numbers, Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT at cover price. AUSTRALIA: Back numbers are obtainable from HOME COMPUTER ADVANCED COURSE. Back numbers, Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G Melbourne, Vic 3001. SOUTH AFRICA, NEW ZEALAND, EUROPE & MALTA: Back numbers are available at cover price from your newsagent. In case of difficulty write to the address in your country given for binders. South African readers should add sales tax.

How to obtain binders for HOME COMPUTER ADVANCED COURSE – UK and Eire: Please send £3.95 per binder if you do not wish to take advantage of our special offer detailed in Issues 5, 6 and 7. EUROPE: Write with remittance of £5.00 per binder (incl. p&p) payable to Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT. MALTA: Binders are obtainable through your local newsagent price £3.95. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Miller (Malta) Ltd, M.A. Vassalli Street, Valletta, Malta. AUSTRALIA: For details of how to obtain your binders see inserts in early issues or write to HOME COMPUTER ADVANCED COURSE BINDERS, First Post Pty Ltd, 23 Chandos Street, St. Leonards, NSW 2065. The binders supplied are those illustrated in the magazine. NEW ZEALAND: Binders are available through your local newsagent or from HOME COMPUTER ADVANCED COURSE BINDERS, Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington. SOUTH AFRICA: Binders are available through any branch of Central Newsagency. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Intermag, PO Box 57394, Springfield 2137.

Note – Binders and back numbers are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK market only and may not necessarily be identical to binders produced for sale outside the UK. Binders and issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.

THE CHAIN GANG



PAUL CHAVE

Interface 1
Networks don't have to involve miles of cables and expensive equipment – Spectrums equipped with Interface 1 can communicate and share microdrives and a printer, making a very low-cost system

Computer networks can be nationwide, like Prestel, or they can be on a smaller scale, linking travel agents with airlines, for example. However, these systems are operated by powerful and expensive mainframe computers. In this article we look at how to set up a network using a group of home computers.

A network is a system of computers linked together to share data and equipment. However, each computer has its own operating system and its own 'protocols' (procedures, formatting rules, and so on) for communicating with the outside world. Because of these problems of compatibility, the individual stations or *nodes* of a network must be similar computers: all Spectrums, or all Apples, for instance.

For the sake of discussion, let us assume that a group of people has five computers that they want to link to a single printer. Our group needs to be able to send information to the printer from any of the nodes. What if two or three nodes have text to be printed out at the same time? And more significantly, what if node 3 has text to be printed out but needs to continue working while the printer is operating? To solve these problems, we have to instal a sixth computer, called the *print server*. This machine is dedicated to controlling the flow of data to the printer and, therefore, cannot be used for anything else. The print server will store the documents in order of priority. Once the piece of text has been sent from a node to the print server, the node can work on other things.

The use of a dedicated machine that acts as a server is essential to a network, because it is through the server that information can be shared. In addition to a print server, some network applications would require a *file server* to handle shared disk drives and to control the flow of information from node to node.

The next step is to create a link among the node computers. This is done by stringing cable — either a twisted-pair or coaxial cable — from machine to machine. Although there are several possible arrangements for the nodes and the server station, including a 'star' and a 'ring', the concept is essentially the same, so we will describe the process in somewhat general terms. Making the connection usually requires a special networking interface for each node. Such an interface might be a simple RS232 connection or a plug-in printed circuit board. In addition, the server station requires a storage unit with enough capacity to handle all the work flow. The server station also requires sufficient RAM to manage the network.

It is the software a computer uses that determines how well a machine performs its function, and this is especially true for networks. First and foremost there must be a 'layer' of software over the machine's operating system that makes the connection between each node and the network. Known as *networking software*, this layer puts the server station in control of the specialised operations of file serving or print serving, and also gives it control of the flow of data within the network. In addition to setting up this chain of command, the networking software informs the computer at each node that it is on a network, that there is a server attached, and how many other nodes there are. Finally, the networking software gives the node computers a protocol for communicating with the rest of the system. This layer of networking software must be up and running for the network to operate.

Once the networking layer is established, the individual nodes must have a program, or a set of programs, for their own applications that recognise the network and know how to communicate with it. This is software written specifically for networking applications, and can be run from a cassette or disk drive at the node, or through the file server. The software is only as complex as the operation. If node 1 in our network is running a word processing program and sending the results to the printer independently of the other nodes, the only modification required to a standard word processor is the inclusion of network protocols. On the other hand, if nodes 2 and 4 need to use the same data, and they need to be able to see each other's results, things become more complicated. In such a case, the applications software (whether it be a word processor, spreadsheet, database, or even a game) and the system hardware must have the ability to do multi-tasking. In other words, the CPU has to be able to handle more than one task simultaneously, and must have the ability to manage communications from at least two other CPUs simultaneously.

TONY SLEEP

Shared Experience
This school was equipped with 16 BBC Micros with colour monitors, a printer, a double disk drive and Econet (Acorn's LAN for the BBC Micro) for £16,000 in 1984 — cheaper than providing each computer with a disk drive and printer. The speed of the network is such that each work station seems to have sole use of the disk drive, even when 30 pupils are at work; terminals may have to join a network queue for the printer. Econet enables David Watkins, teacher in charge of computing, to give all his pupils regular 'hands-on' experience; inter-terminal communication is a valuable bonus when the network is used for subject teaching

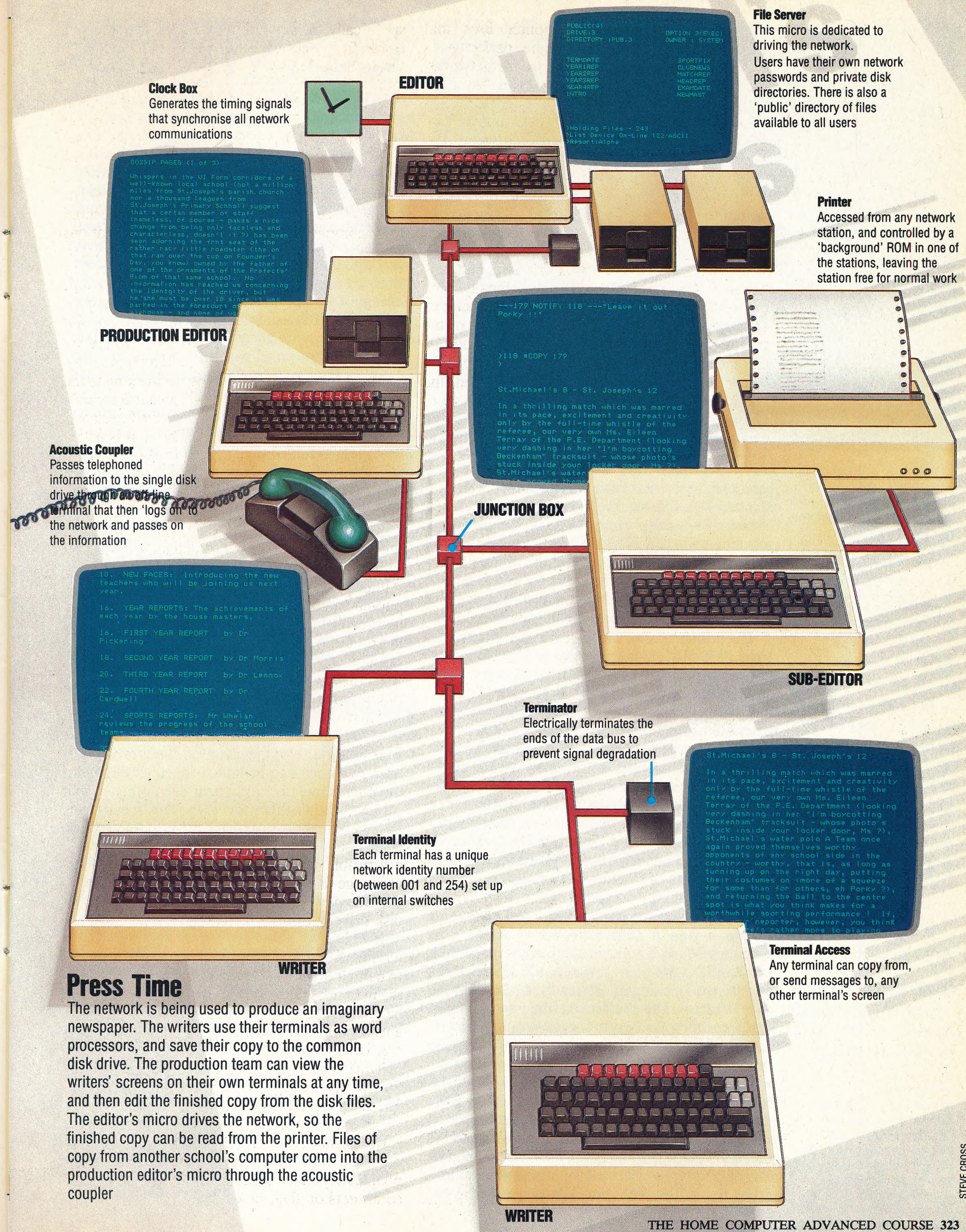
The driving force in bringing networking to the home user seems certain to be Sinclair Research. A networking interface is built into the Interface 1 add-on unit for the Spectrum. This unit is selling well because it is needed to enable Microdrives to be used with the Spectrum. Once sufficient Interface 1s have been sold, software is likely to appear that makes use of the network interface.

Sinclair's latest micro, the QL, has a similar networking interface built in as standard, and this should be compatible with the Spectrum version. Although this interface is fairly crude, the popularity of these machines should make it worthwhile producing networking software for them. Games, programs are the obvious first candidates. Beyond that, the possible applications for home computer users are, unfortunately, rather limited.

There are several elements needed before networking computers becomes truly feasible. The simplest, of course, is that there must be at least two micros to be linked together. Secondly, the micros must be fairly near each other so that cables can link them together. This means they have to be in the same building. Lastly, there has to be enough 'traffic' to make the network practicable. This means it needs users who either exchange data many times a day, or who want to share expensive equipment (such as printers or disk drives) to offset the cost of the network.

If only a small amount of data were moved around the network it would be easier for one user to hand it to another as a tape or disk. Similarly, if the network consists of only a few micros, it could be cheaper to provide each with its own printer and disk drive, rather than investing in the extra cost of computers.

Thus, apart from games, the only practical uses for the networking of home micros are in small businesses and the classroom. The Sinclair QL has cut the cost of networking down to a level where it is worthwhile providing a computer for staff who do not need to use computers heavily. Many people may soon find themselves with networked computers on their desk at work.



Press Time

The network is being used to produce an imaginary newspaper. The writers use their terminals as word processors, and save their copy to the common disk drive. The production team can view the writers' screens on their own terminals at any time, and then edit the finished copy from the disk files. The editor's micro drives the network, so the finished copy can be read from the printer. Files of copy from another school's computer come into the production editor's micro through the acoustic coupler

STEVE CROSS

ALL UNDER CONTROL

ELITE
ABCDEFGHIJKLMNOP
QRSTUVWXYZabcdef
ghijklmnopqrstuv
wxyz0123456789 !
"£$%&'()*+,-./:;
<=>?@[\]^_`{|}~

PICA
ABCDEFGHIJKLM
NOPQRSTUVWXYZ
abcdefghijklm
nopqrstuvwxyz
0123456789 !"
£$%&'()*+,-./
:;<=>?@[\]^_`
{|}~

EMPHASISED PICA ITALIC
ABCDEFGHIJKLM
NOPQRSTUVWXYZ
abcdefghijklm
nopqrstuvwxyz
0123456789 !"
#$%&'()+,-./*
:;<=>?@[¥]^_`
{|}~

ENLARGED ELITE
ABCDEFGH
IJKLMNOP
QRSTUVWX
YZabcdef
ghijklmn
opqrstuv
wxyz0123
456789 !
"£$%&'()
*+,-./:;
<=>?@[\]
^_`{|}~

DOUBLE-STRIKE CONDENSED PICA
ABCDEFGHIJKLMNOPQRSTUVW
XYZabcdefghijklmnopqrst
uvwxyz0123456789 !"£$%&
'()*+,-./:;<=>?@[\]^_`{
|}~

Dotted Around
Dot matrix printers offer a range of typefaces such as Pica, Elite, and Italic, and typestyles such as condensed, enlarged and emphasised. All the examples shown here were produced by the Epson FX-80

Even the cheapest dot matrix printers incorporate a range of 'special effects', such as the ability to print in large characters, which can make a print-out a far more exciting document visually. Here, we show you how such effects are obtained — and how to get your computer and printer 'talking' to each other in the first place!

A dot matrix printer can do far more than simply produce program listings. A quick leaf through the pages of the printer's user manual will show you that a variety of 'special effects' can be produced on paper. Even the cheapest dot matrix printers will let you alter the size of the characters printed on the paper. Normally, the text is printed out at 80 characters per line, but this number can be increased by selecting the 'condensed print' mode (which uses smaller characters), or decreased by selecting 'enlarged print'. In a similar way, the line spacing — the gap between the lines of text — can be altered. A large spacing given by four lines per inch, for example, could be reduced to, say, eight lines per inch, giving a heavier density print-out.

The printer that we will look at in detail here, the Epson FX-80, is a fine example of a machine that has a wide range of printing features. The emphasised mode, which prints out text in darker type, and the alternative mode, which switches from the normal typeface to *italic* characters, are two of its standard facilities. But perhaps its most interesting feature is its ability to change any of the characters stored in the printer's memory, an extremely useful facility for foreign alphabets or for printing scientific symbols. Before going on to investigate how these features are produced, however, let's consider how a printer goes about the simple task of printing out a program listing.

The way that a computer 'talks' to a printer varies from machine to machine. The Dragon, for example, uses a simple variation of the LIST command — LLIST — to instruct the printer to produce a copy of a program. Other machines require the opening of 'channels' or 'streams' to gain access to the printer. As the exact method varies so much, it is best to consult your computer's user manual — the printer manual is unlikely to be of much use here.

Having established communication between the two machines, your first print-out may be a little disappointing. The most likely problems are that all the text has been printed out in one indecipherable black line, or there are blank lines between each line of the program. The explanation for both these faults lies in the difference between a 'line feed' character and a 'carriage return' character. After your computer has sent a line of text to the printer, it also sends a carriage return character, which moves the print head back to the left margin ready to print a new line. Some computers also send a line feed character to move the paper up one line; others assume that the printer does this automatically. To further complicate matters, most printers have an internal switch that decides whether the printer generates its own line feeds or not. If either of these problems occurs, find this switch — by consulting the printer manual — and flick it to the alternative position.

Apart from producing program listings, a printer can also be used as an output device — instead of characters being displayed on the screen, they are printed out on paper. Again, the exact method of doing this varies from computer to computer — the 'standard' BASIC command is LPRINT and this is used by the Spectrum and Oric. On a Commodore 64, OPEN1,4 followed by PRINT#1,"HELLO" would print the word 'HELLO'. With a Dragon micro the same task is accomplished using PRINT#—2,"HELLO". The BBC Micro uses VDU2 followed by PRINT "HELLO" and the VDU3 command. The programming examples that we give here use LPRINT, so you might have to alter this for your machine.

ADDRESS LABELS

```
10 LPRINT "MR JOHN SMITH"
20 LPRINT "7 THE PARADE"
30 LPRINT "ANYTOWN"
40 LPRINT "ABC 123"
50 FOR I=1 TO 7
60 LPRINT
70 NEXT I
100 GOTO 10
```

This listing is a simple program to produce address labels. These can be purchased on a roll with sprocket holes on both sides, so that they can be used with the tractor feed on the printer. Because it does not use any special control codes, the program will work with any make of printer. As it stands, the program will print the same name and address repeatedly. You might want to alter it so that you can input different names and addresses, or even have it read them from a data file. The FOR...NEXT loop between lines 50 and 70 prints seven blank lines, and is used to position the print head at the beginning of each label correctly. The exact number of blank lines may need to be adjusted for your machine.

Our program is quite adequate for simply printing labels, but to print something more complex, like an invoice or letterhead, we are going to have to use some of the special effects that we mentioned earlier. These are produced by sending control codes to the printer as well as to the normal text characters.

In addition to having a code for each character on the keyboard, the ASCII character set (see page 77) has a group of 'invisible' characters that do not print anything on the screen or paper. It is these codes that are used to turn on the printer's special effects: in the standard ASCII set there are four codes (17, 18, 19 and 20) that are reserved as device control commands. Unfortunately, the ASCII character set does not have enough reserved control characters for the 70-odd features of an Epson FX-80, and in order to overcome this, most effects are produced by sending 'escape codes' to the printer. These consist of two or more character codes, starting with an ESC character (ASCII code 27). For example, to turn on the proportional spacing feature on an Epson you send ESC-p — i.e. the Escape character followed by the lower-case 'p' character.

In BASIC, this is written as:

```
LPRINT CHR$(27);"p"
```

The ESC character cannot normally be produced by pressing the Escape key on your keyboard, and consequently the CHR$ function is used.

On the BBC, you would use:

```
VDU2
VDU1,27,1,112
VDU3
```

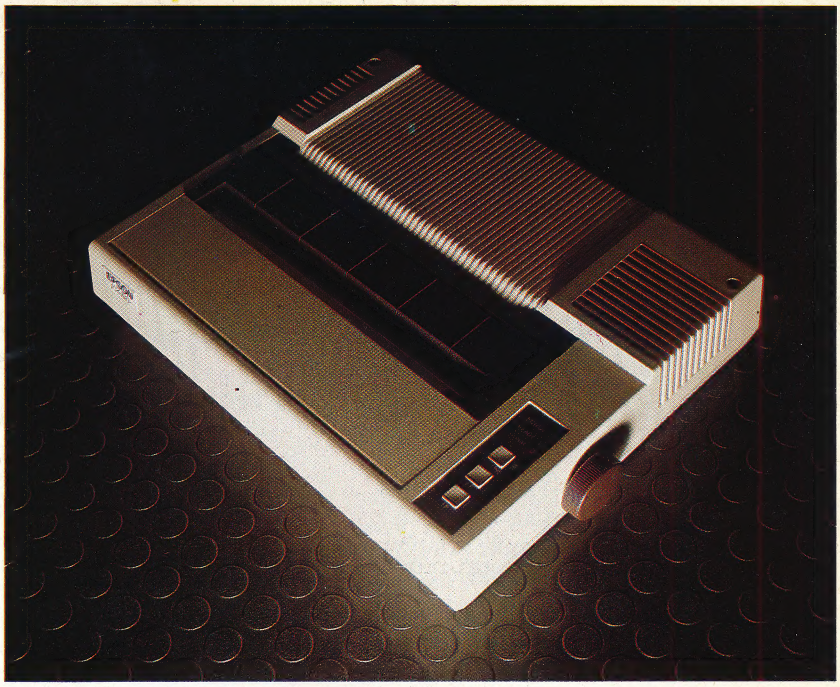
The VDU2 command turns on ('enables') the printer; VDU1 means 'send the following character to the printer only' (PRINT would send the following ESC character to the screen as well, with undesirable results). VDU3 turns off ('disables') the printer.

These command sequences apply only to the Epson FX-80. If you try to send the same code sequence to a different printer, it will either have no effect, do something unexpected, or cause the printer to 'hang up' (i.e. refuse to respond to the computer).

CREATING AN INVOICE

Our second listing demonstrates the use of some of the Epson's features to create an invoice heading, as might be used by a small garage. The codes we have used here are those used by the Epson FX-80. The Epson range of printers is one of the most popular; so much so that other manufacturers make models that are 'Epson-compatible'. If your printer is incompatible with the Epson, however, you must alter the control codes accordingly.

```
999 REM INVOICE HEADING
1000 LPRINT CHR$(12)
1010 LPRINT CHR$(14);TAB(12);"HCAC
MOTORS LTD."
1020 LPRINT CHR$(13);CHR$(13)
1030 LPRINT CHR$(27);"E";
1040 LPRINT TAB(36);
1050 LPRINT CHR$(27);"-";CHR$(1);
1060 LPRINT "INVOICE";
1070 LPRINT CHR$(27);"-";CHR$(0);
1080 LPRINT CHR$(27);"F";
1090 LPRINT CHR$(13);CHR$(13);CHR$(13)
1100 REM INVOICE DETAILS PRINTED
```



IAN McKINNELL

Epson FX-80
A popular printer among business and home micro owners, though expensive at £500. The FX-80 has a nine-pin head, and maximum print speed of 160 characters per second. Most software packages (word processors, for example) support Epson and Epson-type printers

To begin, line 1000 sends the character with code 12 to the printer. This is the 'form feed' character, which instructs the printer to roll the paper to the start of a new sheet. Then we have ASCII code 14; this is called the 'shift out' (SO) character, and on the Epson it causes all subsequent text to be printed in enlarged letters. In our program it is used for the heading, giving the name of the garage in large letters. The TAB function is used to centre the heading.

CHR$(13) is the carriage return character, which produces a single blank line when printed on its own. Several are used in lines 1020 to 1090 to space out the top of the invoice. ESC-E in line 1030 turns on the emphasised mode, and all subsequent text is printed in darker type (caused by printing the same letters several times over). Line 1050 turns on the 'underlining' feature, and line 1070 turns it off, after printing and underlining the word 'Invoice'. ESC-F disables the emphasising mode. The print-out will look like this:

```
HCAC MOTORS LTD.


INVOICE
```

We have shown only the initial part of the program here; a completed invoice program would include lines to print out customer details — name, make of car, money owed, etc. These details would have been obtained from a series of questions at the beginning of the program, and the answers would have been stored as variables.

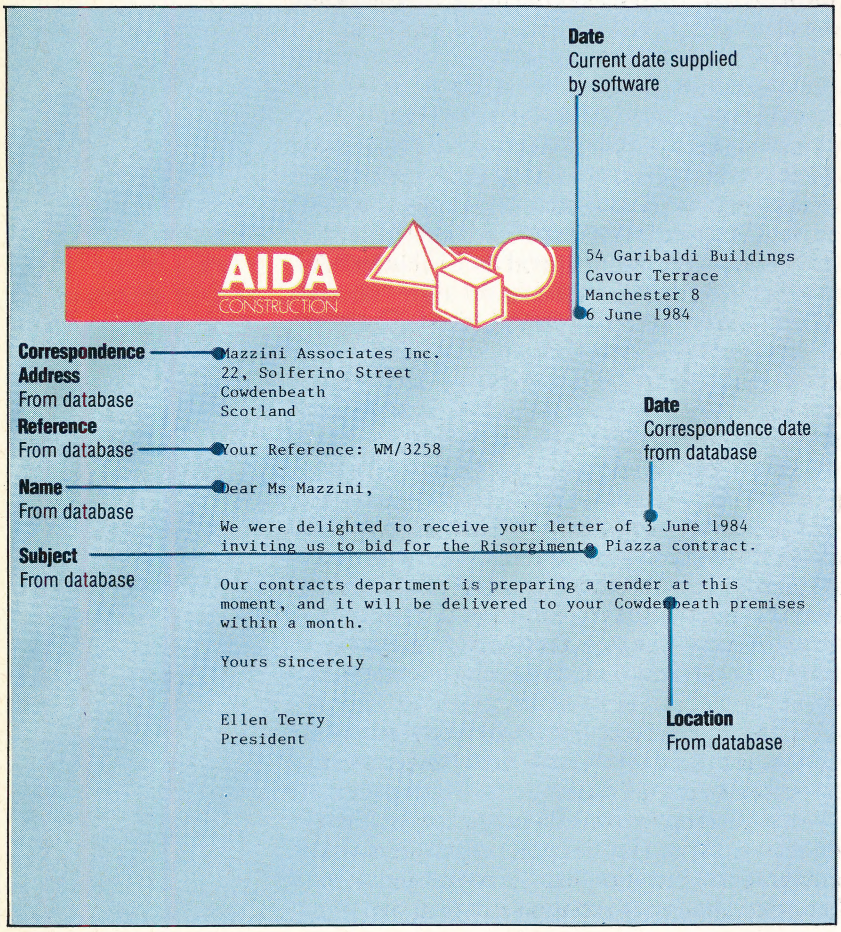
The two programs that we have given here are simple examples of the sorts of alternative uses that a dot matrix printer can be put to. Many people are now exploring the use of a printer beyond simply using it to make program listings. In fact, programming your printer can be just as enjoyable as programming the computer itself.

POST HASTE

One of the most onerous tasks of office workers and club secretaries is having to sit in front of piles of envelopes, writing or typing out addresses. To create an effective mailing system for a home micro all you need is a printer, a word processing package and a database program.

Computerised mailing can go a lot further than the simple printing of address labels. In fact, if all we needed was a way of addressing envelopes, the word processor would not be necessary. A database package would be sufficient, since setting up a name and address file is very simple on a standard database. Although each name and address record would initially have to be input field by field, once the whole system was set up we would be able to use the database package, time after time, to generate as many labels as we needed.

Since database packages, if they are any good, have some sophisticated reporting facilities, it would be possible to select the labels to be printed according to certain criteria: for instance, all addresses with London in the town field. This would take care of selecting and addressing the labels on the envelopes, but it would still leave the user with the other half of a boring job to do. If, for example, the purpose of all the addressing of envelopes was to send a standard letter to a particular set of customers, it would also be necessary to type the addresses on each of the letters.



Furthermore, each customer would have his own address and name on the letter, but the text of the letter itself would remain anonymous and impersonal. There is little point, for example, in leaving gaps in the text to be filled in later with each customer's name, since names differ in length. You would have to leave gaps large enough for the longest name you wanted to include, and the 'personalised' letter that resulted would look phoney. So personalising the letter would mean retyping it as many times as there are names on the mailing list.

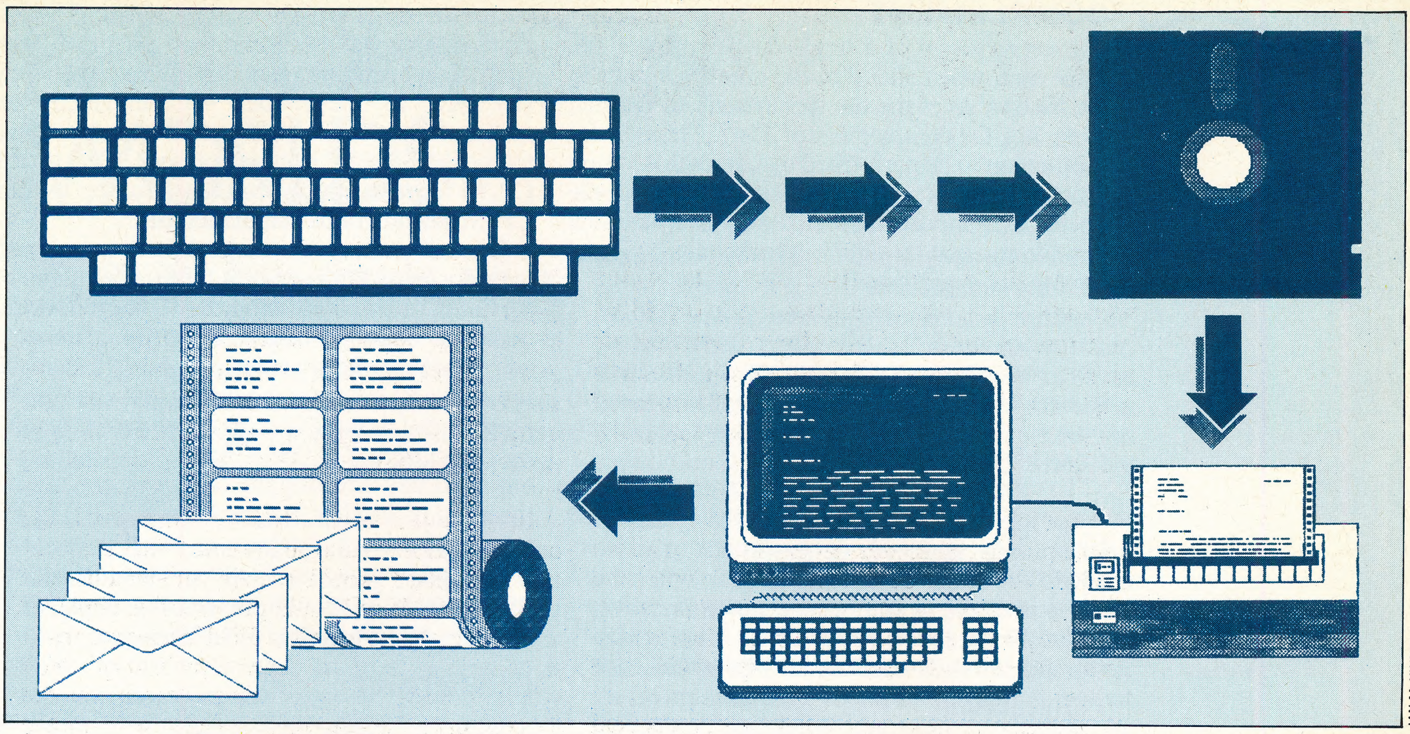
The ideal solution to this problem is to arrange the system so that the user can compose a standard letter using all the facilities of a word processor, and then have the computer automatically pull the relevant details for each customer off a database file to add personalised details to each individual letter.

There are a number of ways of doing this. One of the simplest is that taken by two disk-based packages produced by Acorn — Memoplan and Fileplan, a word processor and a database respectively. Both of these packages come as part of a selection of software given away by Acorn to purchasers of its Z80 second processor for the BBC Model B micro.

The packages cannot produce personalised letters independently and must be used together. The name and address details have to be set up in Fileplan and the standard letter (or form) is created using Memoplan. Each field in the name and address file is numbered, and at relevant points in the standard letter the user simply types in the number of the field whose contents need to be transferred to the letter. The computer will then work through the mailing list sequentially and the contents of the field numbers for all the relevant records on the file will appear in the standard letter.

The word processor automatically adjusts the surrounding text so that the recipients receive a letter that looks as if it has been personally written to them. Memoplan also allows the person setting up the standard letter to include a reminder to help identify the contents of a particular field. For example, 2(SURNAME) indicates that the second field contains the surnames of all recipients.

Specialised mailing packages aimed at business



microcomputers like the IBM PC and the Sirius, such as Micropro's Mailmerge (the best-known mailing program) or Peachtree's Mailing List Manager, include a number of helpful and very sophisticated features. Not only can a large number of different address files be created, but they can also include comprehensive search and select facilities that enable special one-off mailings of parts of a list. For example, a golf club secretary using such a system would have no trouble mailing all new golf club members with a handicap of less than 15 who have paid their fees.

Search and select is achieved through the standard database techniques of indexing and keying records, then carrying out logical tests of selected fields in the record to see if they fulfil the user-specified conditions for inclusion in the list.

These advanced packages also have detailed formatting facilities. This is a particularly useful feature, since the names and addresses produced by the system can be tailored to fit the size and shape of the labels. With the Peachtree system, for example, selecting the LABEL FORMAT option on the main menu produces the image of a box on the screen, together with a listing of all the fields on the mailing list record. This gives the user a visual representation of the label being created. There are also facilities for informing the printer that, for instance, the label stationery is three labels wide, so three labels should be printed with each pass of the print head, and so on.

In general, mailing programs are at their most useful when dealing with sizeable lists. If you have only a handful of labels to do, it is probably easier to type them individually – unless, of course, the information is already on computer! Because of this, mailing programs for home computers (even when they are ROM-based) generally require you to store your data on disk rather than on cassette.

One example of a package designed for the standard BBC Model B is that provided by GCC, a software company based in Cambridge. GCC has, for some time, marketed its own ROM database called Starbase, which had all the necessary facilities for database—only mailing lists. But the company has now released a version of Starbase in 16K ROM, which provides full mailing facilities in conjunction with the ROM–based word processing package, Wordwise.

Starbase comes with a manual and a utilities disk. In conjunction with Wordwise, its mailing facilities include personalising of standard letters and label formatting. The utilities disk makes it possible to send commands to the printer while running the label format option. So a selection of different typefaces may be used on machines with suitable printers.

One particularly useful feature of Starbase as a mailing package is its ability to carry out arithmetical operations on fields in an address file. In this way, documents such as personalised statements and invoices for club membership can be created, with the computer carrying out all the necessary individual calculations.

Commodore 64 users have a good choice of mailing packages. One example is Visawrite, produced by Visa Software. Unlike Starbase, Visawrite will work with either a cassette or disk, and can be used with any database package that can create a sequential file. On the other hand, used as a processing package on its own, it will hold up to 500 names and addresses. These are set up by using each document page as a record card and merging them with a standard letter.

When used on its own, Visawrite does not have selective search facilities, although users can browse through names and addresses and mark them individually for inclusion in a mail run.

Chain Mail
The word processor supplies a skeleton letter with blanks where specific information such as date, name and address should be, and the database supplies the relevant data from its records. After the letters, addresses are printed on sticky labels

Mailing List Software Prices	
Memoplan	Free with Z80
Fileplan	Free with Z80
Mailmerge	£167
Mailing List Manager	£202
Starbase	£69
Wordwise	£46
Visawrite	£80 (Disk)
	£90 (ROM)
All prices include VAT at 15 per cent	

COMPLEMENT

The *complement* of a single-digit number is the value that, when added to the number, results in the base value of the number system. In decimal (base 10), the complement of 3 is 7. This enables subtraction to be performed by using the addition function and a couple of other very simple operators. Suppose in decimal we wish to subtract three from eight (8—3=?). An alternative way of writing the expression is:

$$(8 + (10 - 3)) - 10 = ?$$

This may seem more complicated, but it isn't. Seven (10 — 3) is the complement of three, and to subtract 10 from the final result, we need only strike off the leftmost digit.

To complement in binary, the obvious approach is to swap all ones and zeros, so that 1011 becomes 0100. This is called *one's complement.* More commonly used, however, is *two's complement,* which consists of a one's complement with one added to the result, so that 1011 becomes 0101. Let's try our initial subtraction problem in binary:

Three is 0011
Two's complement is thus 1101
Eight is 1000
Adding these numbers gives 10101
Deleting the leftmost digit gives five (0101).

CONCATENATE

The dictionary definition of the verb *concatenate* is 'to join or link together, especially in the form of a chain'. In programming terms it means the same, and is usually applied either to alphanumeric strings in RAM or to files on a disk. The assignment statement A$ = B$ + C$ is one form of concatenation. To concatenate two or more files on a disk means to link them into one larger file, with a single new filename.

CONCURRENCY

A *concurrent* operating system on a microcomputer is one that allows several applications or utility programs to be run simultaneously on the same processor. In practice, this means switching rapidly and automatically between tasks, which is why the technique is also known as *multi-tasking* or *time slicing.* Though multi-tasking has been common on mainframe computers for many years, full scale concurrency has yet to be widely implemented on microcomputers — which, until recently, have featured relatively slow processors. Concurrent CP/M from Digital Research will run up to four CP/M-86 programs simultaneously.

Since there is only one keyboard, screen and operator in such a system, it follows that three of the tasks must be autonomous — they have no need for operator intervention. By pressing a couple of keys, the operator can select which of the four tasks is shown on the screen and can start or finish any of them. The most sophisticated feature of Concurrent CP/M is known as *piping* — automatically directing the output of one task to the input of another. An application of this might consist of using a database to select names and addresses from a file, feeding these to a word processing program and using a third program to print the reports in the background.

Multi-Tasking
Concurrent CP/M is available on many business micros, including the IBM PC. Fast disk drives, a 16-bit processor and large memory are essential when four separate tasks are concurrent

CONSTANT

Unlike a variable, which is a character or string used to label an item stored in the computer that may change its value during a program run, a *constant* is any item (string or numeric) that remains unchanged. In the program statement:

```
10 A = B*6.5731
```

A and B are variables, while 6.5731 is a constant. This should be familiar ground to anyone who has mastered BASIC, but many users are not aware of the implications of using constants or variables.

Constants should be used sparingly because they have several drawbacks, and should be replaced by variables wherever possible. For a start, they slow down calculations. This is because most computers perform arithmetic in floating point binary format, not in binary coded decimal (see page 168). Each time BASIC encounters a line such as the example given, it must convert the constants into floating point format, whereas the variables are already stored in this format.

Another drawback is that constants waste memory if the same value is used several times in a program. It is better to assign the constant to a single-letter variable at the start and use that instead. Programs using variables are easier to edit, as all constant values may be found in a block of assignment statements at the start of the program, making changes much easier.

CONTENTS ADDRESSABLE

This is a method of addressing a memory location by referring to the data stored there rather than by specifying the address itself. This is particularly useful in database applications, providing a search mechanism for a chosen data item. For example, a file of names, addresses and telephone numbers could be searched for occurrences of dialling code '01'; the result will be a list of names and addresses of people living in London.

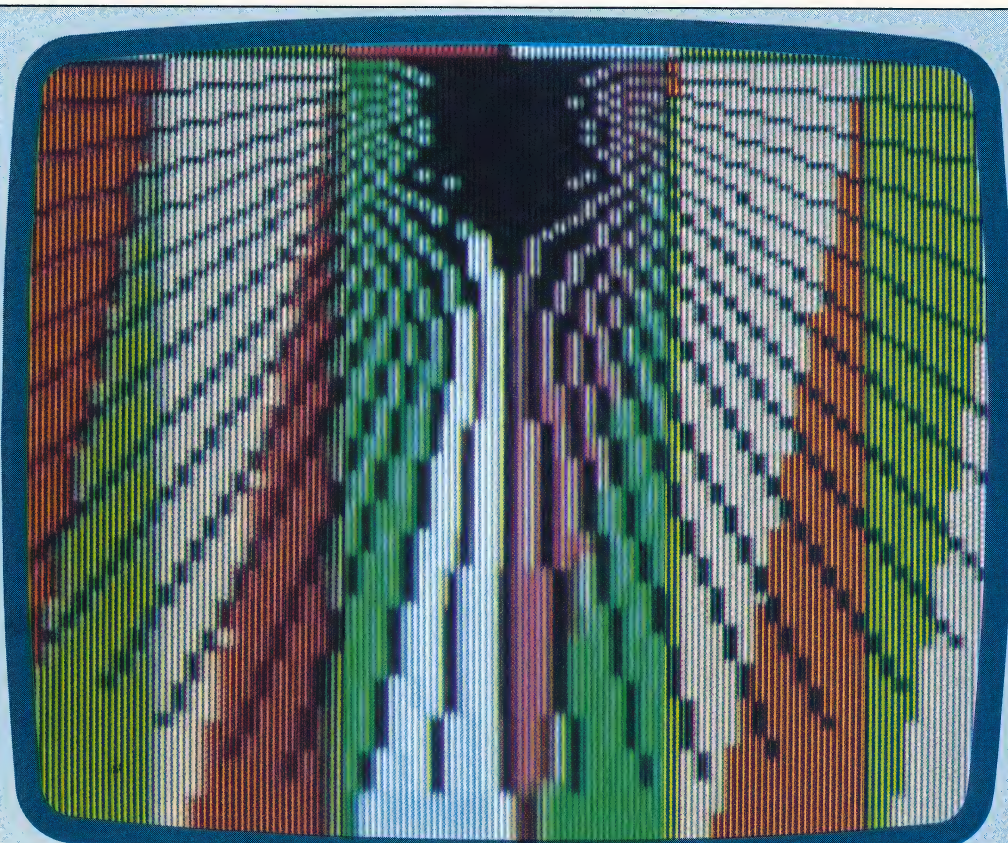
BEST OF BOTH WORLDS

Although any home computer may be connected to a television set, the display quality is much improved if a monitor is used. Here we examine a third alternative — the use of a combined television receiver and monitor to give a high-quality display coupled with the ability to receive television transmissions.

Most home computer owners soon become accustomed to the wavering pictures and indistinct colours produced by their machines when connected to an ordinary television set. For those fortunate enough to have access to a monitor, however, the improved picture quality comes as a revelation — colours are clear and distinct, the whole display is steadier, and there is a marked absence of the 'dot crawl' that plagues television users ('dot crawl' is the shimmering effect that is particularly noticeable on the edges of text displayed on the screen). But there is a price to pay for this higher quality — monitors are more expensive, and cannot be used to receive television programmes.

Now, however, there is a third choice: the combined television/monitor gives users the best of both worlds. It comprises an ordinary television receiver with an additional socket to give monitor-quality when connected to a microcomputer. Some users may already possess one of these hybrids without realising it, as many of the newer television sets are equipped with sockets for connection to video recorders, and these are equally suitable for micro use.

The problems associated with the use of ordinary televisions as computer displays stem from the way they receive signals. Television programmes are transmitted in the form of radio waves; these are picked up by the television aerial and converted into pictures. A home computer simply mimics this process by passing its output through a modulator (the small box inside the computer to which the aerial lead connects). After the modulator has altered the signal to the 'radio wave' form acceptable to the television, the receiver then changes it again to produce a display. This means that the signal can be corrupted in two places — at the modulator and inside the receiver itself. A monitor dispenses with the modulation; it runs directly from the raw picture signal, giving a high-quality display.



IAN McKINNELL

Three Degrees

There are three main types of display signal produced by computers. All home computers produce television signals but this often means a poor picture. Most computers also produce signals for monitors. These give a better display but are expensive. TV/monitors combine the quality of a monitor with the ability to pick up broadcast television pictures. The main difference in quality between TV sets and monitors is the type of signal they work from. These three images were produced on the same TV/monitor but using the three different types of signal, television (sometimes called RF), composite video and RGB. The television signal (shown in the middle) gives the poorest quality, the composite video (on the left) is a little better and the RGB signal (on the right) gives the best result

One factor to be considered when purchasing a monitor is the format used by your computer. There are two types of monitor signal in common use: RGB (Red, Green, Blue) and composite video. RGB gives a better picture, but both types are considerably superior to television output.

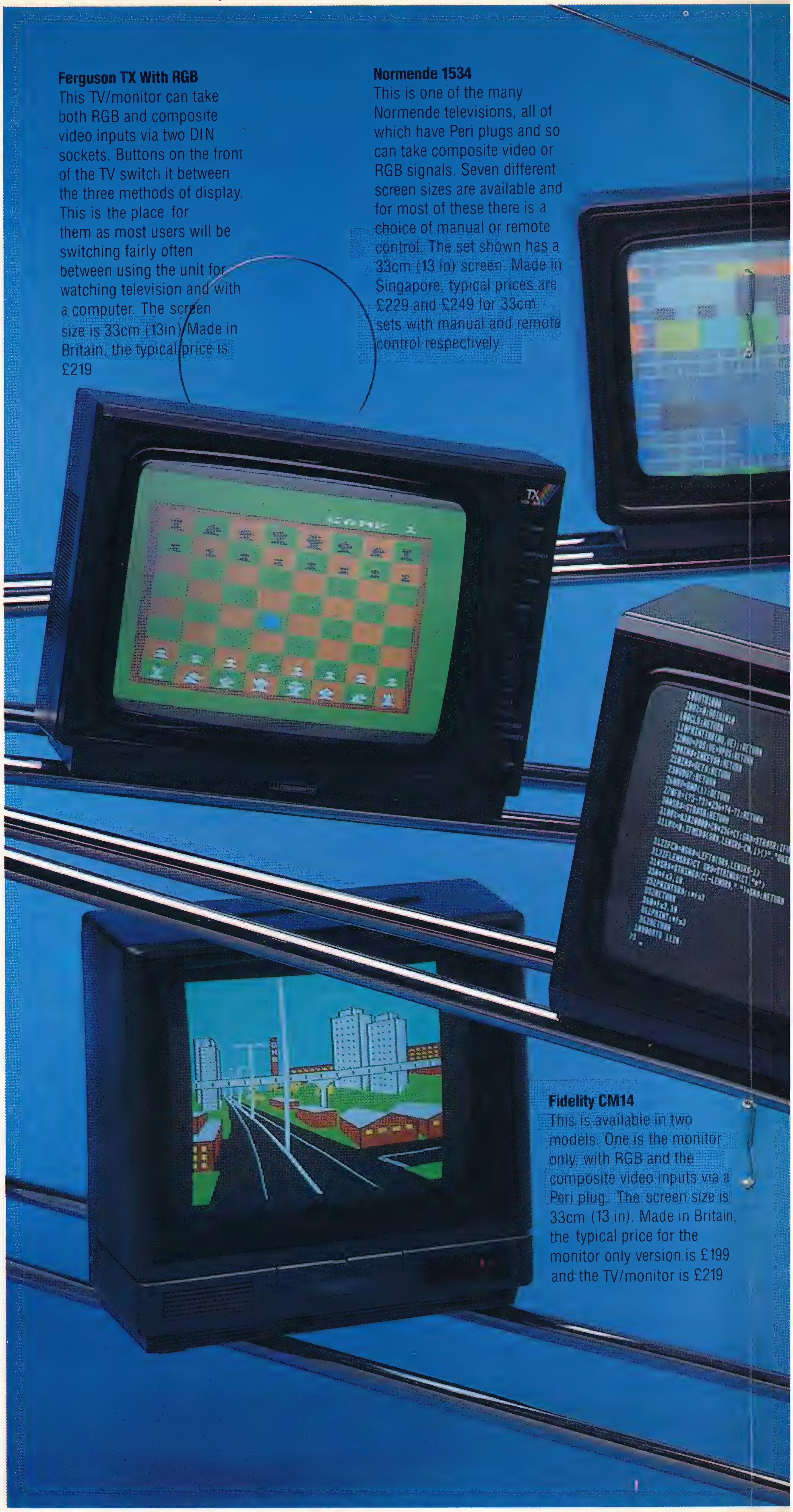
There are also two types of television/monitor — ordinary television receivers that have been converted to take a monitor signal, and purpose-built sets. The latter are more suitable, as converted sets are often modified without the television manufacturers' knowledge and thus will probably not be covered by a guarantee. Purpose-built television/monitors are mainly designed for use with video recorders. These generally feature composite video inputs — look for a socket marked 'video' or 'audio-visual'. The diagrams accompanying this article will show you how to connect your computer (assuming you have a composite video model) to one of these sockets. Once this is done, you may tune your set to the computer's display in the same way as you would select a television channel.

The major advantage of a combined television/monitor over a standard monitor is the sound facility. Many home computers — notably the Atari, Commodore and Dragon models — rely on the television set to produce sound effects. A standard monitor has no sound facilities, while television/monitors have built-in loudspeakers and amplifiers.

If your computer is equipped with RGB output, your choice is more limited. There are three main RGB-input television/monitors: the Sony Profeel system, televisions with Peri-TV connectors (notably the Normende range) and the ITT model. The Sony Profeel accepts both RGB and composite video signals, but uses a non-standard connector. The ITT television/monitor has an RGB connector that is pin-for-pin compatible with Oric and Atmos outputs but which may also be used with other RGB computers. The Normende is especially popular with home computer owners, as it features a Peri-TV socket. This is an international standard television expander socket that will accept both RGB and composite video signals.

Other television sets may be fitted with Peri-TV (also known as 'Scart') inputs — check to see if yours is one of these. The only problem with this system is that, on some sets, switching from television to monitor mode is accomplished by insertion and removal of the Peri-TV plug. This is much less convenient than selecting the computer display by switching channels, as you'd do on a receiver with a video socket. The fact that the Peri-TV system is compatible with both RGB and composite video inputs means that you can keep the same television/monitor, even if you change your computer.

But regardless of the particular system chosen, the superior performance of a combined television/monitor should make this the only type of television set a computer user should ever buy.



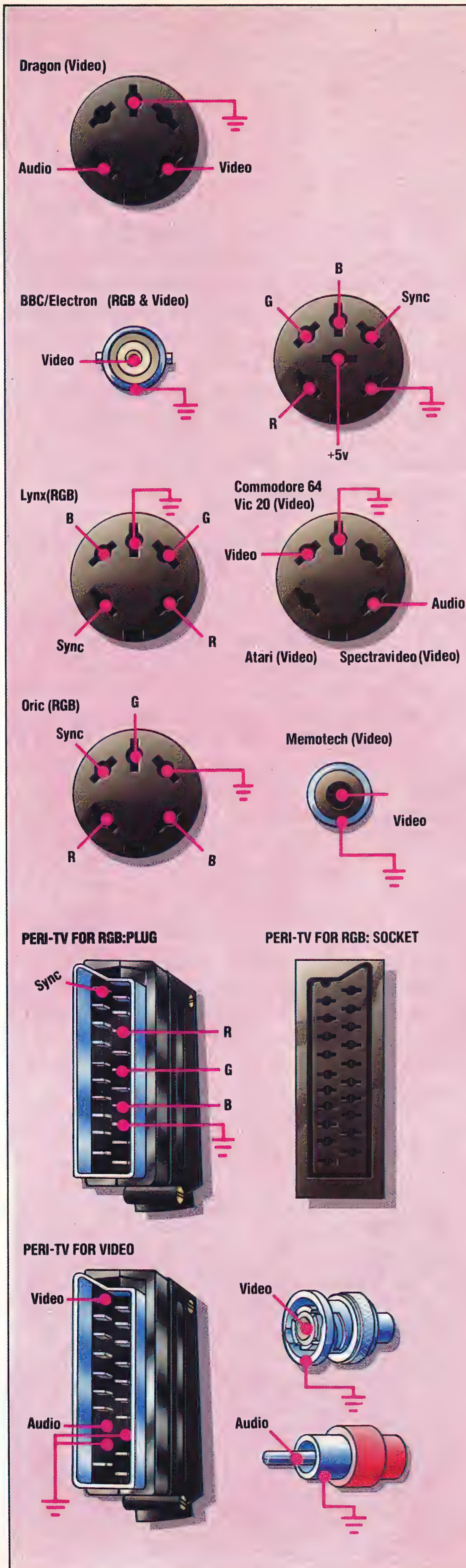
Ferguson TX With RGB
This TV/monitor can take both RGB and composite video inputs via two DIN sockets. Buttons on the front of the TV switch it between the three methods of display. This is the place for them as most users will be switching fairly often between using the unit for watching television and with a computer. The screen size is 33cm (13in). Made in Britain, the typical price is £219

Normende 1534
This is one of the many Normende televisions, all of which have Peri plugs and so can take composite video or RGB signals. Seven different screen sizes are available and for most of these there is a choice of manual or remote control. The set shown has a 33cm (13 in) screen. Made in Singapore, typical prices are £229 and £249 for 33cm sets with manual and remote control respectively

Fidelity CM14
This is available in two models. One is the monitor only, with RGB and the composite video inputs via a Peri plug. The screen size is 33cm (13 in). Made in Britain, the typical price for the monitor only version is £199 and the TV/monitor is £219

ITT RL2315
The ITT has both composite video and RGB inputs. These are via two DIN sockets. A switch on the back of the monitor selects which type of signal it uses, but this is slightly awkward to get at. Screen size is 33cm (13 in). Made in Britain, the typical price is £199

CHRIS STEVENS



Making Connections

There are two different types of monitor signal: composite video and RGB. RGB signals consist of separate red, green and blue signals plus a 'sync' signal. RGB monitor outputs and inputs use multipin (usually DIN) sockets. A composite video signal has all the colour signals and the sync combined into just one signal. Composite video input and output sockets are usually phono or BNC (bayonet-type) sockets. However, some computers include the sound output from the same socket as the video signal; in these cases, multipins must be used.

The diagrams show the connections to be found on home computers. If your television/monitor has one of the common connections shown, all you have to do is make up a cable with the two plugs, according to the connections given — e.g. R to R, sync to sync, and so on.

Television/monitors with Peri-TV sockets require the switching from television to monitor mode to be done by the plug. These usually require a five volt output from the computer (as the BBC has) and some form of switching circuit like the one shown.

Two important computers, the Sinclair Spectrum and ZX81, are missing from the list because they lack monitor interfaces, although it is possible to modify them to give a composite video signal. At least one company produces an adaptor for the Spectrum to give a better quality RGB signal. This adaptor plugs into the micro and so does not invalidate the guarantee.

The Commodore Vic 20, Atari, Spectravideo and early versions of the Commodore 64 all use the same input for composite video. Recent models of the Commodore, however, use an eight-pin DIN plug. These should be wired with pin two as earth, pin three as sound and pin four as video

KEVIN JONES

MAKING WAVES

For most microcomputer users, mathematics is a boring subject that has little relevance outside the classroom. But mathematical formulae are very important in the production of computer graphics. Here we explain how to build up three-dimensional graphs by entering different mathematical functions into a program.

The very mention of sines and tangents, let alone graphs in three planes, is enough to cause fear in those glad to have escaped from school mathematics. Yet with the aid of a computer these subjects can become enjoyable, even if the principles behind them are not fully understood.

The graphic abilities of most microcomputers make them ideal for displaying graphs of mathematical equations. Most of us will find such equations meaningless when they are written out as mathematical symbols, but they produce attractive patterns when plotted in the form of a graph. Even those who hate maths may be inspired to produce their own equations after seeing these displays.

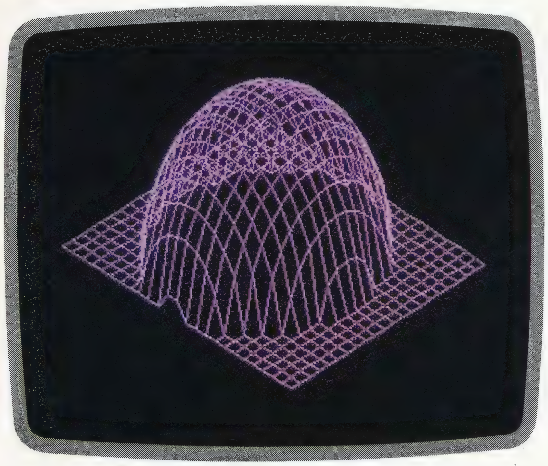
All the patterns shown here were produced on a home micro using the programs listed. They are all calculated as graphs in three dimensions. Everybody knows what an ordinary two-dimensional graph looks like. A 3-D graph is composed of several two-dimensional graphs displayed at the same time, with slight differences between each one. As computers can display images in two dimensions only, the result is not truly three-dimensional, but an illusion of depth is obtained by the way the images are formed.

The programs listed here calculate the values of an equation with two variables, X and Z. The result, Y, is calculated for many values of X and Z. Each value of Y is used to plot a point on the screen, with values of Y corresponding to points on the vertical axis — i.e. the higher the value of Y, the nearer the top of the screen the point will be plotted. Neighbouring points are joined together with straight lines, giving a curved effect. The curves in one direction represent graphs of X and Y, with Z held constant, while the curves that intersect these are graphs of Y and Z, with X held constant (in this case they are plotted on the plane with axes Y and Z, which is at right angles to the X-Y plane of normal two-dimensional graphs). Such displays are useful in helping to understand complicated functions.

These displays can also be a stimulus to people who don't usually take much interest in mathematics. It's fun (and quite difficult) to attempt to come up with an equation that results in a particular shape. To alter the displayed graph, it is necessary to change the function in line 170 of the BASIC program. Some functions may be relatively complicated, and may thus require more than one program line; if this is the case then all lines between 151 and 179 may be used.

In addition to choosing a function that results in a pleasing shape, you must take care that the values produced are not so large that the graph extends beyond the screen boundaries. To keep the display within bounds, the function may need to be divided by a large number.

Versions of this program will work on several home computers. As an aid to conversion, we have designed the program so that the first section sets up the screen display in a standard way. This means that an equation that works on one particular machine should also work on others. The second part of the program is used to store the values for plotting points on the graph. These



```
165 C=60-X*X-Z*Z
170 Y=SQR(C*(SGN(C)+1))/45
```

Creating Shapes
Modify the program as shown under each photograph to produce these 3D graphs

```
170 Y=(SIN(X)+COS(Z))/60
```

results are held in an array and take time to work out. The calculations depend on the function chosen and may take several minutes; during this

170 Y=SIN(X+Z)/12

period the computer appears to be doing nothing. Calculating the function first saves time in the long run. If calculations were made while the lines were being plotted, the program would take almost twice as long to plot the graph.

We have listed several different functions for you to experiment with. The illustrations show the results you can expect. You should also try to develop your own graphs by entering different functions into the program. Take care when doing this; you must make sure that the graph will fit on the screen, and that no illegal mathematical operations are attempted. The two most common errors are trying to divide by zero (which gives infinity) and attempting to find the square root of a negative number (there is no such thing).

To avoid division by zero, add a very small constant (say 0.00001) to any variable that might become zero. The only way to protect against square roots of negative numbers is to use the ABS function to make all the numbers positive before finding the square root.

Some interesting displays may be produced by common mathematical functions such as SIN, COS, LOG, etc. Others may be achieved by using

165 C=X*X+Z*Z+0.00001
170 Y=SGN(INT(23/C))/3+SGN(INT(55/C))/15

functions that are found only on computers — try INT, SGN and ABS.

This program may be improved in a number of ways. You could try adapting it so that any function is automatically scaled to fit within the screen boundaries, or you could try plotting points in a third direction, giving curves for X and Z while Y is kept as a constant (this is relatively complicated). But even if you simply use the program as it is written here, you should find that it is amusing to try out the silliest equations you can think of. The results may surprise you.

This program is written in BBC BASIC.

```
 10 REM * GRAPH PLOTTING
 20:
 30 REM * SET UP SCREEN
 40 ACROSS=1280:TALL=1024 :UP=-1
 50 XGAP=25:ZGAP=15
 60 WIDE=INT(ACROSS/XGAP/2)
 70 DEPTH=INT(TALL/ZGAP/3)
 80 MODE4:CLS:PRINTTAB(12)"CALCULATING"
 90:
100 REM * CALCULATE GRAPH
110 START=20
120 DIM G(WIDE,DEPTH)
130 FOR A=-DEPTH/2 TO DEPTH/2
140 FOR B=-WIDE/2 TO WIDE/2
150 X=A*20/WIDE:Z=B*20/DEPTH
160 REM * INSERT FUNCTION BELOW HERE
170 Y=(SIN(X)+COS(Z))/60
180 G(B+WIDE/2,A+DEPTH/2)=Y*UP*TALL
190 NEXT B:NEXT A:CLS
200:
210 REM * DRAW GRAPH ; X-Y PLANE
220 FOR Z=1 TO DEPTH
230 XBASE=XGAP*Z
240 ZBASE=TALL/2+Z*ZGAP+START*UP
250 XOLD=XBASE+XGAP
260 ZOLD=ZBASE-ZGAP-G(1,Z)
270 FOR X=1 TO WIDE
280 XNEW =XBASE+X*XGAP
290 ZNEW=ZBASE-X*ZGAP-G(X,Z)
300 PLOT 4,XOLD,ZOLD:PLOT 5,XNEW,ZNEW
310 XOLD=XNEW:ZOLD=ZNEW
320 NEXT X:NEXT Z
330:
340 REM * DRAW GRAPH ; Z-Y PLANE
350 FOR X=1 TO WIDE
360 XBASE=XGAP*X+DEPTH*XGAP
370 ZBASE=TALL/2-X*ZGAP+DEPTH*ZGAP+START*UP
380 ZOLD=ZBASE-ZGAP-G(X,DEPTH-1)
390 XOLD=XBASE-XGAP
400 FOR Z=0  TO DEPTH-1
410 XNEW=XBASE-Z*XGAP
420 ZNEW=ZBASE-Z*ZGAP-G(X,DEPTH-Z)
430 PLOT 4,XOLD,ZOLD:PLOT 5,XNEW,ZNEW
440 XOLD=XNEW:ZOLD=ZNEW
450 NEXT Z:NEXT X
460:
470 REM * HOLD DISPLAY
480 GOTO 470
```

Basic Flavours

Spectrum

Insert LET in all assignment statements. Insert the following lines:

```
 40 LET ACROSS=256:LET TALL=176:LET UP=-1
 50 LET XGAP=5:LET ZGAP=3
 80 CLS
290 PLOT XOLD,ZOLD : DRAW XNEW-XOLD,ZNEW-ZOLD
410 PLOT XOLD,ZOLD : DRAW XNEW-XOLD,ZNEW-ZOLD
```

Oric-1/Atmos

Insert the following lines:

```
 40 ACROSS=239:TALL=199:UP=1
 50 XGAP=5:ZGAP=3
 80 HIRES
300 CURSET XOLD,ZOLD,1:DRAW XNEW-XOLD,ZNEW-ZOLD,1
430 CURSET XOLD,ZOLD,1:DRAW XNEW-XOLD,ZNEW-ZOLD,1
```

TRICKS OF THE TRADE

Most people teach themselves programming by using the manual that comes with their computer. This is a good enough way to get started, but it often means you never learn to write efficient programs nor discover the tricks of the trade that make programming easier. We introduce a series of articles designed to give you insights into the techniques used by good programmers.

Good programming is developed through experimentation and experience. The novice programmer, often solving problems through enormous enthusiasm and sheer effort, is gradually transformed into a technician with an awareness of short-cuts and rule-of-thumb methods that achieve the desired results. Eventually, the programmer will develop the simple clarity and direct approach of the expert. But there is no reason why the personal progress of a home computer programmer cannot be hastened by learning from the mistakes of others who have taken the same path. The lessons are there for the learning, and everyone's programming can benefit from them. Our course begins with a discussion of some of the more helpful hints that can aid a beginner.

Programming is a problem-solving process, and a great part of it should be carried out in the mind and with a pencil and paper long before a line of code is written. The stages in this process are well-known: a clear comprehensive statement of the problem in practical terms, followed by repeated re-statement of the problem with increasing precision, until it is formulated with as much detail and accuracy as possible. This description nearly always contains or implies the essential solution, which must then be expounded in greater and more practical detail so that it becomes a working method. In programming, only the last stage should involve coding, and that should be a straightforward realisation of the preceding stages. When the coding stage overlaps the real problem-solving, poor solutions and bad code result.

Solutions are often known as *algorithms*, processes of computation analysed in logical stages. The efficiency of a program depends mainly upon that of its algorithm, and this is judged in terms of its 'completeness' and its 'correctness'. These two commonsense qualities refer to the program's theoretical and practical ability to cope with the foreseeable range of input conditions, and to the consistency of its internal logic. Needless to say, it's much easier to recognise their absence than to demonstrate their presence, but every program must be subjected to this judgement, and the earlier in its development the better.

Solutions must be *reliable*, as well as complete and correct. Not only must they handle their prescribed range of problems, but they must also deal predictably and safely with conditions outside their range. This usually means having the ability to recognise potential error conditions, and being able to stop operating with all the data intact, as well as displaying some useful status message. It is difficult to judge whether code is sufficiently reliable, as a program that isn't reliable is easier to recognise than one that is. Experience leads to better judgement.

Making programs reliable and robust is a worthy aim that nearly always conflicts directly with an equally desirable goal — keeping them *economical*. Everything costs money, even if it's only the time you spend writing programs for fun. There always comes a moment when you have to decide between continuing to work on a program that's nearly 'bombproof', and abandoning it to start a fresh project. Even if your time is unlimited, the computer's memory and operating speed are not. It's quite possible to surround the central algorithm with so much precautionary code and error-trapping that protecting against crashes can take more time than solving the original problem.

TESTING AND DEBUGGING

Solving analytical and logical problems in theory is enormously important, but programs are meant to perform a task. Once the first syntax and logical errors have been dealt with it's time to begin *testing*. This is so familiar an idea that it hardly seems to merit statement, never mind emphasis. But it is, in fact, a much misunderstood process. In anything but trivial programs there are usually far too many possible combinations of input conditions for exhaustive trials, so tests must be devised to put as much strain as possible on what are likely to be the most vulnerable (and what are expected to be the strongest) parts of the program. Generating comprehensive test conditions is not a simple matter and takes time and money. The professional approach to testing is that there are no perfect programs, only bad tests.

Successful tests reveal a program's inadequacies, and should do so in a logical fashion so that *debugging* takes as little time as possible. Like testing, debugging is an essential process that regularly fails to be achieved precisely because it embodies the same human failings that make it necessary in the first place. A program bug should be approached as another problem to be solved, exactly as described earlier — statement, analysis, algorithm, testing — but it is most often treated as a casual pest to be swatted, poisoned or crushed, with predictably disastrous consequences for its surroundings.

As these development stages are completed, so familiarity and satisfaction combine to convince the programmer that the program works now, will always work and will never need changing, and anyway the code is a model of clarity. But programmers, not programs, need documentation. No program is self-explanatory, and there are always reasons for wanting to change working programs. Like any other mechanism, they need *maintenance*, and maintenance means manuals. Programs should be internally documented (using REM lines) for the programmer's benefit, and externally documented with accompanying literature for the sake of the user — even if the user is the programmer.

All of these lessons once had to be learned by mainframe programmers, and have been ignored and painfully rediscovered by microcomputer programmers. Taken together they comprise a programming 'structure', a unified approach to problem-solving far more comprehensive than a book of cautionary tales about avoiding GOTOs or embracing WHILE . . . WEND. Efficient programs are written by efficient programmers on a basis of structured experience and logical thinking. This series of articles aims to encourage both.

IAN McKINNELL

Bar Charts
The colours and depths of the bars forming the chart are easily adjusted in the program by changing the values of the control variables

```
399 REM***************************
400 REM*   3-D BAR CHARTS        *
401 REM***************************
500 GOSUB 1500:        REM INITIALISE
720 YY=22:XX=2:        REM ORIGIN COORDS
760 GOSUB 3200:        REM DRAW AXES
800 FOR E=LT-1 TO 0 STEP -1
820 OC=XX+E*DB:OL=YY-E*DB
840 GOSUB 2200:        REM INPUT DATA
900 FOR D=1 TO NN
920 HT=DT(D)
940 X0=OC+(D-1)*(BB+LT*DB):Y0=OL-HT-DB
960 GOSUB 4000:        REM PRINT BAR
980 NEXT D
1000 NEXT E
1100 XP=10:YP=23:GOSUB 3500
1120 PRINT"THREE-DIMENSIONAL HISTOGRAM"
1200 A$=INKEY$:IF A$="" THEN GOTO 1200
1400 END
1499 REM***************************
1500 REM*  INITIALISE  S/R        *
1501 REM***************************
1520 CL$=CHR$(147):  REM CLEAR SCREEN
1540 PRINT CL$
1560 PO$=CHR$(19):   REM HOME CRSR
1580 RT$=CHR$(13):   REM <RETURN>
1600 BB=2:DB=1:      REM BAR DIMENSIONS
1620 SW=40:SD=25:    REM SCREEN DIM'S
1640 HB=SD-DB:       REM MAX BAR HEIGHT
1660 DIM B$(HB+DB)
1680 FOR K=1 TO SD:PO$=PO$+RT$:NEXT K
1800 DIM DT(SW)
1900 GOSUB 2400:     REM BUILD BAR
2100 LT=4:           REM DEPTH FACTOR
2190 RETURN
2199 REM***************************
2200 REM* INPUT DATA ARRAY S/R    *
2201 REM***************************
2220 READ NN
2240 FOR Z=1 TO NN:READ DT(Z):NEXT Z
2310 DATA 6,12,10,4,7,8,10
2320 DATA 5,7 ,8 ,8,6,7
2330 DATA 6,7 ,4 ,8,5,3,9
2340 DATA 5,11,6 ,4,11,6
2390 RETURN
2399 REM***************************
2400 REM* BUILD WHOLE BAR S/R     *
2401 REM***************************
2500 TC$=CHR$(158):  REM SIDES=YELLOW
2520 FC$=CHR$(31):   REM FRONT=BLUE
2540 RV$=CHR$(18):   REM REVERSE ON
2560 NL$=CHR$(146):  REM REVERSE OFF
2580 CR$=CHR$(29):   REM CURSOR RIGHT
2600 CH$=CHR$(32):   REM SPACE CHAR.
2620 C1$=CHR$(169):  REM "◤"  CHAR.
2640 C2$=RV$+C1$:    REM "◢"  CHAR.
2680 FOR K=1 TO SW
2700 SP$=SP$+CH$
2720 RC$=RC$+CR$
2740 FF$=FF$+CH$
2760 NEXT K
2800 TL$=SP$+"/"
2820 BL$=SP$+NL$+C1$
2840 FL$=LEFT$(FF$,BB)
2900 L$=TC$+C2$+RV$+RIGHT$(TL$,BB)
2920 FOR K=1 TO DB
2940 B$(K)=LEFT$(RC$,DB-K)+L$+LEFT$(SP$,K-1)
2960 NEXT K
3000 L$=FC$+RV$+FL$+TC$+RIGHT$(TL$,DB)
3020 FOR K=DB+1 TO HB
3040 B$(K)=L$
3060 NEXT K
3100 L$=FC$+RV$+FL$+TC$
3120 FOR K=1 TO DB
3140 B$(HB+K)=L$+RIGHT$(BL$,DB+2-K)
3160 NEXT K
3190 RETURN
3199 REM***************************
3200 REM*    DRAW   AXES    S/R   *
3201 REM***************************
3300 PRINT TC$:          REM SET COLOUR
3320 FOR Y=2 TO YY-1
3340 XP=XX-1:YP=Y:GOSUB 3500:PRINT"!"
3360 XP=XX+YY-Y:GOSUB 3500:PRINT"/"
3380 NEXT Y
3400 YP=YY
3420 FOR X=XX-1 TO SW-1
3440 XP=X:GOSUB 3500:PRINT"-"
3460 NEXT X
3490 RETURN
3499 REM***************************
3500 REM* PUT CRSR  @ XP,YP S/R  *
3501 REM***************************
3600 PRINT LEFT$(PO$,YP)TAB(XP-1);
3620 RETURN
3999 REM***************************
4000 REM* PRINT PARTIAL BAR S/R  *
4001 REM***************************
4100 FOR V=1 TO HT
4120 XP=X0:YP=Y0+V-1
4140 GOSUB 3500:      REM PLACE CRSR.
4160 PRINT B$(V)
4180 NEXT V
4200 FOR V=1 TO DB
4220 XP=X0:YP=Y0+HT+V-1:GOSUB 3500
4240 PRINT B$(HB+V)
4260 NEXT V
4490 RETURN
READY.
```

Structure
Modular, and reasonably self-explanatory

Documentation
This routine is obviously crucial, but is entirely unexplained

Variable Names
Well commented, but not very meaningful

Completeness/Correctness
Does this work for all values? Does it produce the right output in all cases — when a value is less than zero, for example?

Error Trapping
No checksum, no scaling, nor error handling

Documentation
Good: no 'magic numbers', no mysterious control characters, everything translatable

Curate's Egg

This program to display three-dimensional bar charts is an annoying mixture of good and bad style: the internal documentation is good where it exists and the structure is modular, but the program is not self-explanatory, there is no error-trapping, and no user documentation

Basic Flavours

This program is written in Microsoft BASIC, and should run unchanged on micros with a 40×25 screen display. The screen dimensions are initialised in line 1620. The control character values initialised in subroutines 1500 and 2400 are for the Commodore 64; consult the ASCII chart in your manual for other machines

VALLEY OF THE TROLLS

Adventure gaming allows the player to perform heroic deeds in fantastic surroundings. Like a crossword puzzle, an adventure is really a battle of wits between the writer and the solution-seeker, and players often spend weeks grappling with a particularly difficult problem. Here we look at Bug-Byte's Twin Kingdom Valley.

Adventure games on computers originally derived from the Dungeons and Dragons role-playing game. In the 1960s, mainframe programmers began developing the first computer versions, using the large amounts of available memory to store details of a complex fantasy world full of wizards and monsters, dwarfs and trolls. Today's microcomputer adventures are all descended from these early examples, but are set in a much wider variety of locations — ranging from abandoned spaceships to the streets of Chicago in the gangster era of the 1930s.

But all the good adventures have one thing in common; the player must be made to feel that the fantasy world is real. The best adventures are almost like novels, with the player becoming totally involved in the situations depicted. Originally, all adventures were text-only, but the new breed use high resolution graphics to bring an added sense of realism to the games. The first big-selling graphic adventure was The Hobbit, based on J. R. R. Tolkien's book of the same name. The adventure we examine here, Twin Kingdom Valley, uses graphics in a traditional adventure setting of mediaeval castles and forests.

While graphics may add a certain gloss to an adventure, it must be said that they cannot disguise a lack of imagination on the programmer's part — and this is the case with Twin Kingdom Valley. The story behind the game is very simple. The player takes the role of a wanderer who ventures into the valley, which is ruled by two warring kings (the Desert King and the Woodland King). In the valley are several rivers (all of which look remarkably similar) that flow into the magical lake Watersmeet. While roaming the Kingdom, the player – a somewhat mercenary hero – must collect as much treasure as possible. When enough has been accumulated, and the player's score has reached 1,024, something surprising – we won''t spoil the game by revealing what this is – happens.

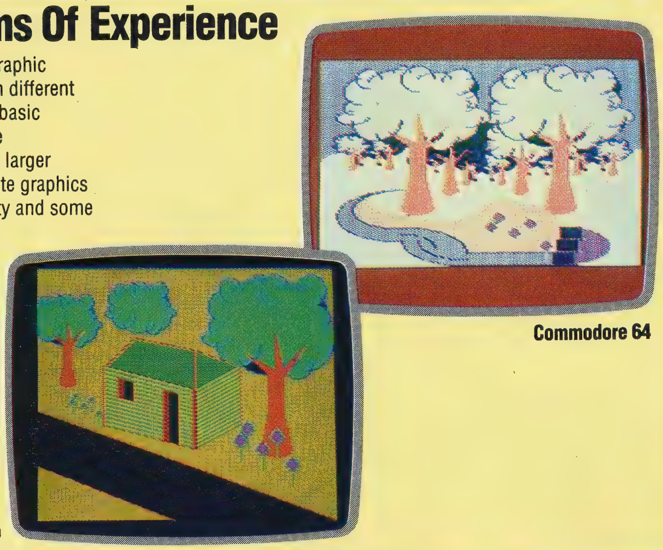
Movement and actions are controlled by typing in instructions. The program accepts 23 verbs, which are combined with nouns referring to objects in the game. An instruction such as 'Hit guard with hammer' will be accepted, always assuming that you have a hammer in your possession and there is a guard within range. Directions are indicated by points of the compass, plus the words 'up' and 'down'. Other characters populate the Kingdom, and you may use the word 'ask' to try to acquire their possessions. In most cases, however, you will be met with unprovoked violence if you attempt to talk to them.

Graphics are used to illustrate 175 of the game's locations. The BBC Micro, in particular, uses a large amount of memory to produce high resolution displays, so most of the pictures are composed of different combinations of the same basic shapes; for example, a forest comprises 10 or 12 tree shapes repeated in various patterns. The Commodore's larger memory and sprite graphics allow animation in some screens, with squirrels climbing trees and water dripping from stalactites. The graphics may be switched off, but still use memory space that could have been better used to make the adventure more exciting.

Twin Kingdom Valley is only moderately difficult to solve, and is hardly original in concept. There are many other text-only adventures that are far more complex and which give the player a much greater sense of involvement in the world that they depict.

Kingdoms Of Experience

The BBC Micro graphic displays are often different combinations of basic shapes, while the Commodore 64's larger memory and sprite graphics allow more variety and some animation



Commodore 64

BBC Micro

IAN McKINNELL

Twin Kingdom Valley: For BBC Micro, £9.50
For Commodore 64, £9.50
Publishers: Bug-Byte Software, Mulberry House, Canning Place, Liverpool L1 8JB
Author: Trevor Hall
Joysticks: Not required
Format: Cassette

PIXEL PLOTTING

We begin a series of articles exploring graphics applications using 6502 and Z80 machine code on the more popular home computers. Here, we discuss the use of 6502 Assembly language to access the Commodore 64's high resolution screen.

The various steps and procedures involved in Commodore high resolution graphics have been thoroughly explored on page 254: we must first switch the Video Interface Chip (VIC) to high resolution mode, and change the character set base address pointer; the block of eight Kbytes starting at location 8192 that will hold the screen memory map must be cleared; and the normal screen memory map (locations 1024 to 2033 — $0400 to $07E7), which is now to be used to hold screen colour information, must be initialised. This last task can be complicated in multi-colour displays, but is straightforward here since we shall have only one background colour on the screen.

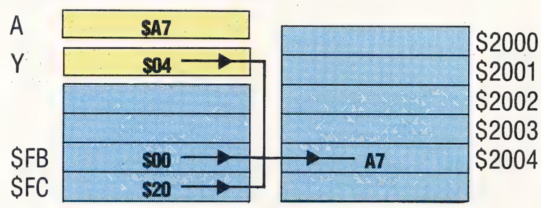
The program will allow switching into and out of high resolution mode, and then perform all the necessary calculations to plot a point on the high resolution screen. So the first part of our program will concern itself with the two important tasks that have to be carried out before the high resolution screen can be used: the colour information has to be put into each of the normal screen locations, and the eight Kbyte bit map has to be cleared.

To allow the same routine to be called when entering or leaving high resolution mode, a flag called HRSFLG will be used. In addition, we may not always wish to clear the bit map on entering high resolution mode, especially if we wish to leave a previously drawn shape on the screen. In order to signal whether we wish to clear the screen, we will make use of a second flag called CLRFLG. The flowchart shows how these two flags will be used within the machine code routine.

Let's now consider the relatively straightforward task of accessing memory blocks of 256 bytes or less using a machine code loop. The following piece of code places the number $03 into each location from the BASE address to BASE+255 (a total of 256 locations in all) using absolute indexed addressing (see page 196).

```
     LDY $00
     LDA $03
NEXT STA BASE,Y
     DEY
     BNE NEXT
```

Note that using this technique BASE is accessed first, but the rest of the block is accessed from BASE+255 downwards to BASE+1. Our program calls for us to access more than one 256-byte block of memory, and in order to do this we must use another form of addressing known as post-indexed indirect. This method uses the zero page to calculate addresses anywhere in memory. Most of the zero page is used by the Commodore 64's operating system, but there are a few free bytes set aside for use by the machine code programmer. Two such bytes are 251 and 252 ($FB and $FC). In this method of addressing, the computer assumes that the lo-byte of the address is held in the zero page location specified and the hi-byte is held in the next zero page location. Thus, an instruction such as STA ($FB),Y, where $FB and $FC contain $00 and $20, and Y contains $04, calculates the required address as follows:



STA ($FB),Y
$2000 + $04 = $2004

In order to access an entire 256-byte block of memory, a similar method to the one just described could be used. The power of this method of addressing lies in our ability to access the hi-byte of the BASE address. Incrementing this hi-byte by one means that the BASE address has actually been increased by 256 (i.e. to the start of the next block of memory). We can apply this technique to the tasks of placing colour information into the normal screen area and clearing the bit map area.

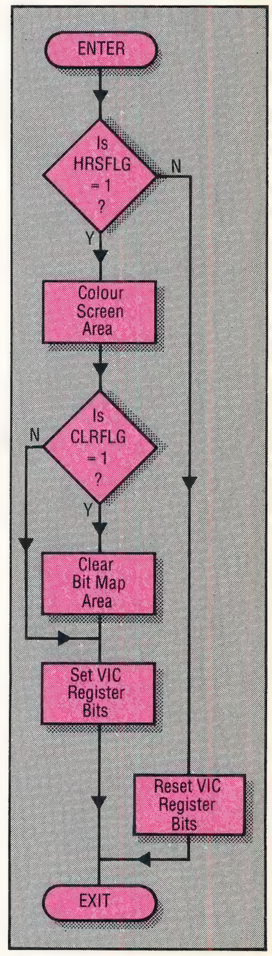
The screen area runs from $0400 to $07E7. This means that it consists of three blocks of 256-bytes and a remainder of $E7 bytes. The block of Assembly language labelled "Colour Screen Area" on page 339 makes use of post-indexed indirect addressing to place the colour information into each byte. Use is made of the variables SCBLO and SCBHI — the lo- and hi-bytes of the normal screen start address — and SCBLK and SCREM — the number of 256-byte blocks and the remainder, respectively. PTR is the zero page location used to store the lo-byte of the base address.

CALCULATING PIXEL POSITIONS

The second part of our machine code routine is concerned with the calculation of the bit in the bit map area corresponding to a given (x,y) co-ordinate. For the high resolution screen,

Hi-Res Routine

This performs three different tasks. It sets up the colour screen information, clears the bit map area, and sets and resets the hi-res register bits – dependent on the state of the flags, HRSFLG and CLRFLG



co- ordinates have x values in the range 0 to 319, and y values ranging from 0 to 199. One byte is sufficient to hold all the possible values of y, but two bytes will be required to store values of x greater than 255. In BASIC, given the co-ordinates x and y, the corresponding bit is calculated using the following steps:

```
1000 HB=XAND248:VBYTE=INT(Y/8)
1010 RMY=YAND7:RMX=XAND7
1020 ROW=VBYTE*320+HB
1030 BYTE=BASE+ROW +RMY
1040 POKEBYTE,PEEK(BYTE)OR(2↑(7-RMX))
```

The corresponding machine code routine has to perform the same calculations. HB, ROW, BASE and BYTE all require two bytes, which makes the arithmetic processes more complex. Most of the coding is self-explanatory, but two sections of interest are those where y is divided by eight and VBYTE is multiplied by 320. Division by powers of two can be easily accomplished:

```
45       = 00101101
45/2 = 22 = 00010110
22/2 = 11 = 00001011
11/2 =  5 = 00000101
5/2  =  2 = 00000010
2/2  =  1 = 00000001
1/2  =  0 = 00000000
```

Each time a division by two is performed, the bits that go to make up the number being divided all move one place to the right. Hence division by eight can be achieved by three Logical Shifts Right (LSR). As we only need the integer part of the answer, we can conveniently ignore any bits that 'drop off' the right-hand end of the number. The LSR operation places the bit that is lost in the carry, so we could use it if we wished. Multiplication by two can, not surprisingly, be done by shifting one place to the left (ASL). We can use this fact to create a routine that will multiply VBYTE by 320. We can think of 320 as 5×64, and 64 itself as 2×2×2×2×2×2. Thus, if we take VBYTE, add it to itself five times and then perform six ASL's we will have effectively multiplied by 320. The only snag is that the answer may well be bigger than 255 and hence two bytes will be needed.

Finally, the two routines we have investigated are both called from within a BASIC program using SYS, and it is important that when the machine code routines have ended, the BASIC program can continue. During execution of a BASIC program, the interpreter makes use of the X, Y and A registers. As these registers are used by the machine code routines as well, it is important to store their contents on entering machine code and restore them on leaving the routine. The most convenient method of doing this is to use the stack. The values of the Y, X and A registers should be pushed onto the stack as soon as the machine code routine is entered and pulled off it before exit. Values for the co-ordinates, colours and flags can be passed to the machine code program by POKEing them into the locations specified, as demonstrated in the BASIC program.

```
1 GOTO 200
2 POKE 53265,PEEK(53265)AND223
3 POKE 53272,PEEK(53272)AND240OR4
4 STOP
200 REM  **** C64 HI-RES DEMO   ****
210 :
220 POKE 56,32:CLR:      REM LWR MEMTOP
250 GOSUB 3000:      REM INITIALISE S/R
350 :
360 REM  **** SET UP HIRES MODE ****
370 :
380 PRINT CC$:PRINT :PRINT
390 INPUT"FOREGROUND COLOUR";FG
400 INPUT"BACKGROUND COLOUR";BG
410 TT=FG*16+BG:     REM CALC. COLOUR
420 POKE COLOUR,TT: REM COL TO M/C S/R
430 POKE HRSFLG,1:  REM SET HIRES ON
440 POKE CLMFLG,1:  REM CLRHIRES SCRN ON
450 SYS BEGIN:      REM ENTER M/C S/R
460 :
500 REM  **** DRAW PATTERN      ****
510 :
515 Z=0:Y1=50:Y2=150:X1=160:SP=6
520 FORY=Y1 TO Y2 STEP SP
530 FORX= Y2-Z TO Y2+Z STEP SP
540 GOSUB1000:      REM PLOT POINT
550 NEXT X
555 Z=Z+SP
557 NEXT Y
560 :
565 GETJ$:IFJ$<>""THEN 565
570 GETA$:IF A$=""THEN 570:REM AWAIT KPRESS
580 :
600 REM  **** CLEAR HIRES SCREEN ****
605 POKE HRSFLG,1:POKE CLMFLG,1
610 SYS BEGIN
620 :
630 REM  ****   LINE DEMO        ****
640 X1=0:X2=300:Y1=0:Y2=190:SP=1
670 GOSUB1500:          REM LINE PLOT
690 :
695 GETJ$:IFJ$<>""THEN695
700 GETA$:IF A$=""THEN700:REM AWAIT KPRESS
710 :
720 REM  ****   RESTORE SCREEN   ****
730 :
740 POKE HRSFLG,0:   REM HRES OFF
750 SYS BEGIN
760 PRINT CC$:PRINT :PRINT
770 PRINTTAB(9)"****END OF PROGRAM****"
780 END
999 :
1000 REM *** HIRES PLOT SUBROUTINE ***
1010 :
1020 XHI=INT(X/HX):XLO=X-XHI*HX
1030 POKE XBYTE,XLO:POKE XPAGE,XHI:POKE YBYTE,Y
1035 SYS PLOT:       REM ENTER PLOT S/R
1040 RETURN
1500 REM *** LINE  PLOT SUBROUTINE ***
1550 C9=(Y2-Y1)/(X2-X1):C8=C9*X1-Y1
1600 FORX=X1 TO X2 STEP SP
1650 Y=X*C9-C8
1700 GOSUB 1000:        REM PLOT POINT
1750 NEXT X
1800 RETURN
3000 REM *** INITIALISE SUBROUTINE ***
3020 CC$=CHR$(147):     REM CLEAR SCRN
3025 HX=256
3030 HRSFLG=49408:      REM $C100
3040 CLMFLG=49409:      REM $C101
3050 COLOUR=49410:      REM $C102
3060 XBYTE =49411:      REM $C103
3070 XPAGE =49412:      REM $C104
3080 YBYTE =49413:      REM $C105
3085 BEGIN =49422:      REM $C10E
3090 PLOT  =49539:      REM $C183
3095 PRINT CC$:PRINT:PRINT
3100 PRINTTAB(9)"****M/CODE LOADER****"
3150 PRINTTAB(9)"1) M/CODE IS ON TAPE"
3200 PRINTTAB(9)"2) M/CODE IS IN DATA"
3250 PRINTTAB(9)"3) M/CODE IS IN MEM."
3300 PRINT"HIT OPTION NUMBER"
3350 FOR LP=0 TO 1 STEP 0
3400 GET OP$
3450 IF OP$>"0" AND OP$<"4" THEN LP=1
3500 NEXT LP
3600 ON VAL(OP$) GOSUB 4000,5000,6000
3900 RETURN
4000 REM** LOAD MCODE FROM TAPE S/R **
4100 PRINT "INSERT TAPE CONTAINING MACHINE CODE S/R"
4200 IF A=0 THEN A=1:LOAD "PLOTSUB.HEX",1,1
```

```
4300 RETURN
5000 REM** LOAD MCODE FROM DATA S/R **
5005 PRINTTAB(11)"*****LOADING*****"
5010 FOR I=HRSFLG TO HRSFLG+312:READ A
5020 POKE I,A:S=S+A:NEXT I
5030 READ CC:IF CC<>S THEN PRINT"CHECKSUM
ERROR":END
5040 DATA 2,0,255,255,2,2,255,255,2,18
5050 DATA 255,255,2,2,72,138,72,152,72
5060 DATA 173,0,193,240,83,169,0,133,251
5070 DATA 169,4,133,252,162,3,160,0,173
5080 DATA 2,193,145,251,136,208,251,230
5090 DATA 252,202,48,8,208,244,145,251
5100 DATA 160,231,208,238,173,1,193,240
5110 DATA 24,169,0,133,251,169,32,133
5120 DATA 252,162,32,160,0,169,0,145,251
5130 DATA 136,208,251,230,252,202,208
5140 DATA 246,173,24,208,41,240,9,8,141
5150 DATA 24,208,173,17,208,9,32,141,17
5160 DATA 208,76,125,193,173,24,208,41
5170 DATA 240,9,4,141,24,208,173,17,208
5180 DATA 41,223,141,17,208,104,168,104
5190 DATA 170,104,96,72,138,72,152,72
5200 DATA 173,4,193,141,7,193,173,3,193
5210 DATA 41,248,141,6,193,173,3,193,41
5220 DATA 7,141,8,193,173,5,193,41,7,141
5230 DATA 10,193,162,3,78,5,193,202,208
5240 DATA 250,173,5,193,141,9,193,169,0
5250 DATA 141,11,193,141,12,193,162,5
5260 DATA 173,11,193,24,109,9,193,141,11
5270 DATA 193,202,208,243,162,6,14,12
5280 DATA 193,14,11,193,144,3,238,12,193
5290 DATA 202,208,242,173,11,193,24,109
5300 DATA 6,193,141,11,193,173,12,193
5310 DATA 109,7,193,141,12,193,173,11
5320 DATA 193,24,105,0,141,11,193,173,12
5330 DATA 193,105,32,141,12,193,173,11
5340 DATA 193,24,109,10,193,141,11,193
5350 DATA 173,12,193,105,0,141,12,193
5360 DATA 173,11,193,133,251,173,12,193
5370 DATA 133,252,169,1,141,13,193,56
5380 DATA 169,7,237,8,193,170,14,13,193
5390 DATA 202,208,250,160,0,177,251,13
5400 DATA 13,193,145,251,76,125,193
5410 DATA 38698:REM*CHECKSUM*
5900 RETURN
6000 REM** MCODE ALREADY IN MEM S/R **
6100 RETURN
```

Using PLOTSUB

The demonstration BASIC program shows the various stages involved in using the machine code high resolution routines:

1) If you have an assembler, you can type in the Assembly language program, assemble it, save it as a source file, then save the object code between $C100 and $C238 under the name "PLOTSUB.HEX". Do not try to execute the subroutine at this stage, since the high resolution screen will probably immediately overwrite the assembler itself, causing it to crash.

2) To use the high resolution routines in a program, you must lower MEMTOP (see program line 220), and load the code from tape (see subroutine 4000).

3) Alternatively, you could save subroutine 5000 as a BASIC program (called "MCODELOAD", say). When you want to use it, lower MEMTOP, then load and run MCODELOAD (thus loading the machine code into memory). Type NEW, then load the program you want to run — the high resolution routines are now in memory and can be accessed by the relevant SYS instructions.

4) The last data item in subroutine 5000 is a checksum — the sum of all the preceding data. If the program stops with a "CHECKSUM ERROR" message, then you have entered the data wrongly, and must correct the mistake before proceeding.

```
;**********************
;**********************
;**                  **
;**  HIRES PLOTTING  **
;**                  **
;**   ROUTINE FOR    **
;**                  **
;**   COMMODORE 64   **
;**                  **
;**********************
;**********************
;
;
PTR    EQU $FB
MPBLO  EQU $00
MPBHI  EQU $20
SCBLO  EQU $00
SCBHI  EQU $04
SCBLK  EQU $03
SCREM  EQU $E7
MPBLK  EQU $20
       ORG $C100

HRSFLG DB  $00
CLRFLG DB  $00
COLOUR DB  $00
XLO    DB  $00
XHI    DB  $00
YLO    DB  $00
HBLO   DB  $00
HBHI   DB  $00
REMX   DB  $00
VBYTE  DB  $00
REMY   DB  $00
ROWLO  DB  $00
ROWHI  DB  $00
BPOS   DB  $00
;++++SWITCH TO HIRES MODE+++++
;++++AND CLEAR BIT MAP AREA+++
       PHA
       TXA
       PHA
       TYA
       PHA
       LDA HRSFLG
       BEQ RESET
;++++COLOUR SCREEN AREA+++++++
       LDA #SCBLO
       STA PTR
       LDA #SCBHI
       STA PTR+1
       LDX #SCBLK
       LDY #$00
       LDA COLOUR
AGAIN  STA (PTR),Y
       DEY
       BNE AGAIN
       INC PTR+1
       DEX
       BMI CLTEST
       BNE AGAIN
       STA (PTR),Y
       LDY #SCREM
       BNE AGAIN
CLTEST LDA CLRFLG
       BEQ HRESON
;++++CLEAR BIT MAP AREA+++++++
       LDA #MPBLO
       STA PTR
       LDA #MPBHI
       STA PTR+1
       LDX #MPBLK
       LDY #$00
       LDA #$00
NEXT   STA (PTR),Y
       DEY
       BNE NEXT
       INC PTR+1
       DEX
       BNE NEXT
;++++SET BIT MAP MODE+++++++++
HRESON LDA $D018
       AND #$F0
       ORA #$08
       STA $D018
       LDA $D011
       ORA #$20
       STA $D011
       JMP EXIT
;++++RESET TO NORMAL SCREEN+++
RESET  LDA $D018
       AND #$F0
       ORA #$04
       STA $D018
       LDA $D011
       AND #$DF
       STA $D011
;++++EXIT M/C+++++++++++++++++
EXIT   PLA
       TAY
       PLA
       TAX
       PLA
       RTS
;++++HIRES PLOT CALCULATION+++
       PHA
       TXA
       PHA
       TYA
       PHA
;++++CALCULATE HORIZ.BYTE+++++
       LDA XHI
       STA HBHI
       LDA XLO
       AND #$F8
       STA HBLO
       LDA XLO
       AND #$07
       STA REMX
;++++CALCULATE VERT.BYTE++++++
       LDA YLO
       AND #$07
       STA REMY
       LDX #$03
SHIFT  LSR YLO
       DEX
       BNE SHIFT
       LDA YLO
       STA VBYTE
;++++CALCULATE ROW++++++++++++
       LDA #$00
       STA ROWLO
       STA ROWHI
       LDX #$05
       LDA ROWLO
       CLC
       ADC VBYTE
       STA ROWLO
       DEX
       BNE FIVE
       LDX #$06
       ASL ROWHI
       ASL ROWLO
       BCC NCARRY
       INC ROWHI
       DEX
       BNE MULT
;++++ADD HORIZONTAL BYTE++++++
       LDA ROWLO
       CLC
       ADC HBLO
       STA ROWLO
       LDA ROWHI
       ADC HBHI
       STA ROWHI
;++++ADD HIRES MAP BASE+++++++
       LDA ROWLO
       CLC
       ADC #MPBLO
       STA ROWLO
       LDA ROWHI
       ADC #MPBHI
       STA ROWHI
;++++ADD REMAINDER OF YLO+++++
       LDA ROWLO
       CLC
       ADC REMY
       STA ROWLO
       LDA ROWHI
       ADC #$00
       STA ROWHI
;++++COMBINE 2 BYTES OF+++++++
;++++ADDRESS ON ZERO PAGE+++++
       LDA ROWLO
       STA PTR
       LDA ROWHI
       STA PTR+1
;++++CALCULATE PIXEL POSN.++++
       LDA #$01
       STA BPOS
       SEC
       LDA #$07
       SBC REMX
       TAX
       ASL BPOS
       DEX
       BNE POWER
;++++TURN ON PIXEL++++++++++++
       LDY #$00
       LDA (PTR),Y
       ORA BPOS
       STA (PTR),Y
       JMP EXIT
```

BEETLEMANIA

Bug-Byte is a software company that has grown in step with the UK home computer industry. It began as a manufacturer of games for the Sinclair ZX80 and has developed into one of the leading suppliers of games for the more popular machines. Its hits have included Manic Miner and Twin Kingdom Valley.

Bug-Byte was founded in the spring of 1980 when Tony Baden and Tony Milner, two chemistry students at University College, Oxford, began writing programs for their newly acquired ZX80. Realising there was little software for the machine, they decided to market their games, bought 40 blank cassettes, and advertised a five-game package in a computer magazine. Orders began to arrive at the rate of 15 a week, and the partners re-invested their profits in more advertisements and writing other ZX80 programs. Later that year the Acorn Atom was launched and Bug-Byte expanded its range to meet the demand for software for the new machine.

In early 1981, the ZX81 appeared and demand for ZX80 games quickly fell, so Baden and Milner turned their attention to writing ZX81 software. They graduated from Oxford in June 1981, and Bug-Byte then moved to Tony Baden's home town of Liverpool. The company now became a full-time operation and within a short time program sales had doubled.

By Christmas 1981, the competition in the software games market had become fiercer. To maintain sales, Bug-Byte employed an advertising agency to handle marketing and started to produce full-colour cassette inserts and advertisements, at the same time using a professional tape duplicating company to ensure high-quality cassettes.

This new approach to the presentation of its products, coupled with the introduction of a nationwide dealer network, had a marked effect on sales and the company employed further staff. As more home computers were launched, Bug-Byte employed freelance programmers to meet the increased demand. Many of these later left to form rival software houses such as Quicksilva and Software Projects. The company pays its programmers at a fixed rate per cassette, and claims that an author whose game reaches the top twenty best-sellers can earn between £10,000 and £40,000 in the first year.

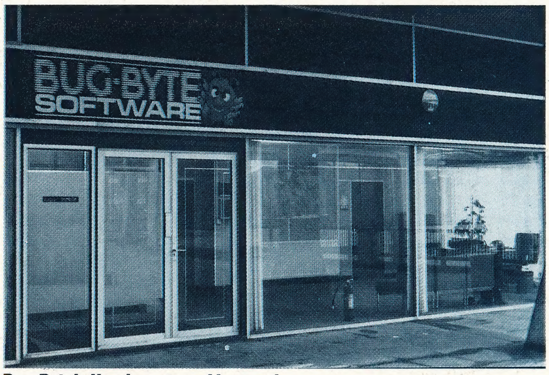
By the end of 1982, Bug-Byte's dealer network comprised over 200 independent outlets, in addition to the major chain stores, and the company's mail order operation was phased out. Tape supplies were proving to be a major headache, as duplicating companies were unable to meet the demand from software houses building up stocks for the Christmas sales boom, so Bug-Byte set up its own duplicating company, Spool. June 1983 saw the company move to larger premises in Canning Place, Liverpool; these had been designed to Bug-Byte's specification and were completed at a cost of £50,000.

Bug-Byte's major successes have included the graphic adventure game Twin Kingdom Valley (for the BBC and Commodore 64), and Manic Miner, which runs on the ZX Spectrum and Commodore 64. The author of Manic Miner, Matthew Smith, has recently left the company to form Software Projects, taking the game's copyright with him.

Bug-Byte's expansion has continued unabated, and the company's software is now sold in most West European countries as well as in Australia, New Zealand and South Africa. A recent deal with CBS UK, which handles Bug-Byte's European marketing, may result in the company entering the lucrative American market. John Phillips, Bug-Byte marketing manager, claims that 'using the CBS connection as a model, we hope to make the operation truly global'.

Tony Milner

Tony Baden



ROGER SANDERS

Bug-Byte's Headquarters, Liverpool

Mentathlete

Home computers. Do they send your brain to sleep – or keep your mind on its toes?

At Sinclair, we're in no doubt. To us, a home computer is a mental gym, as important an aid to mental fitness as a set of weights to a body-builder.

Provided, of course, it offers a whole battery of genuine mental challenges.

The Spectrum does just that.

Its education programs turn boring chores into absorbing contests – not learning to spell 'acquiescent', but rescuing a princess from a sorcerer in colour, sound, and movement!

The arcade games would test an all-night arcade freak – they're very fast, very complex, very stimulating.

And the mind-stretchers are truly fiendish. Adventure games that very few people in the world have cracked. Chess to grand master standards. Flight simulation with a cockpit full of instruments operating independently. Genuine 3D computer design.

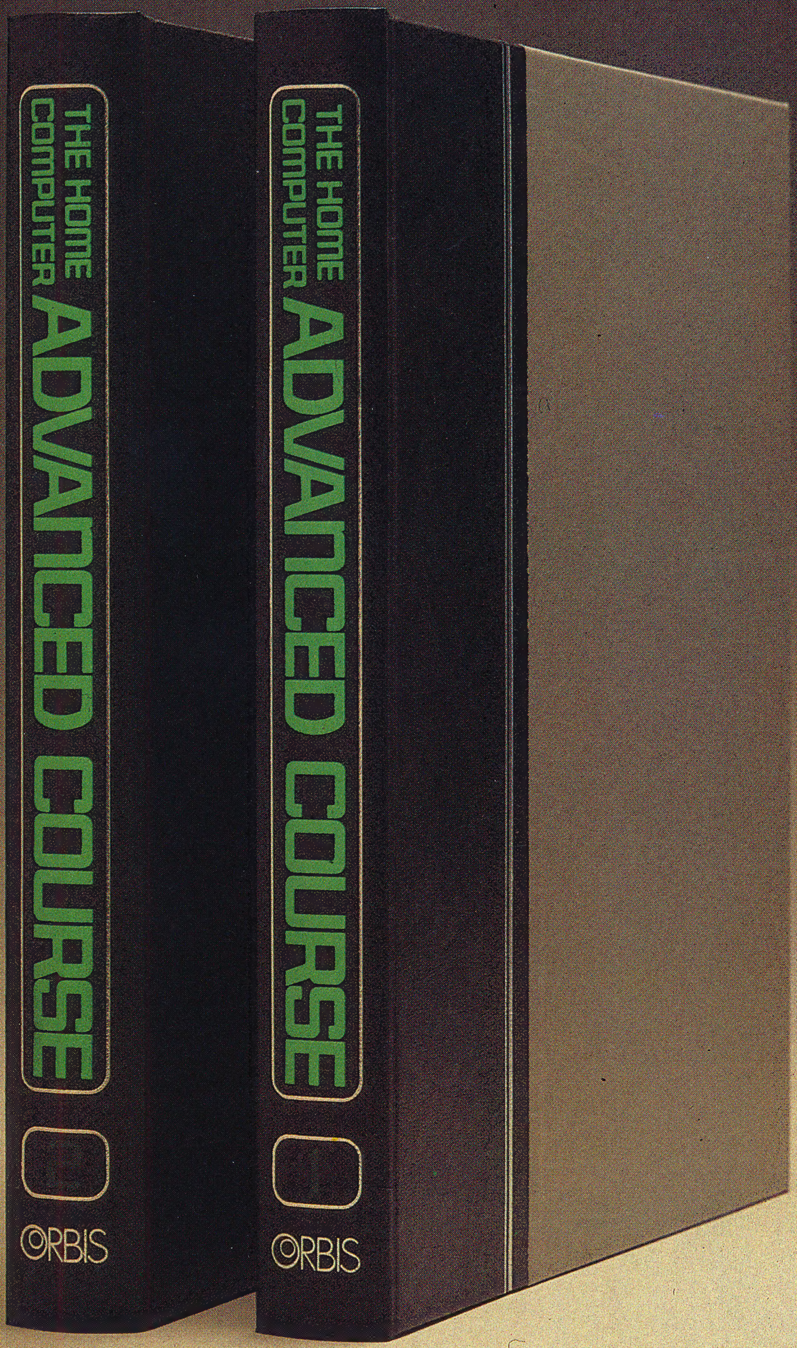
No other home computer in the world can match the Spectrum challenge – because no other computer has so much software of such outstanding quality to run.

For the Mentathletes of today and tomorrow, the Sinclair Spectrum is gym, apparatus and training schedule, in one neat package. And you can buy one for under £100.

sinclair

THE HOME COMPUTER ADVANCED COURSE

WE HAVE DESIGNED BINDERS SPECIALLY TO KEEP YOUR COPIES OF THE 'ADVANCED COURSE' IN GOOD ORDER.



All you have to do is complete the reply-paid order form opposite – tick the box and post the card today – **no stamp necessary!**

By choosing a standing order, you will be sent the first volume free along with the second binder for £3.95. The invoice for this amount will be with the binder. We will then send you your binders every twelve weeks – as you need them.

Important: This offer is open only whilst stocks last and only one free binder may be sent to each purchaser who places a Standing Order. Please allow 28 days for delivery.
Overseas readers: This free binder offer applies to readers in the UK, Eire and Australia only. Readers in Australia should complete the special loose insert in issue 1 and see additional binder information on the inside front cover. Readers in New Zealand and South Africa and some other countries can obtain binders now. For details please see inside the front cover. Binders may be subject to import duty and/or local tax.
The Orbis Guarantee: If you are not entirely satisfied you may return the binder(s) to us within 14 days and cancel your Standing Order. You are then under no obligation to pay and no further binders will be sent except upon request.

PLACE A STANDING ORDER TODAY.